Analysing Supply Chain Operation Dynamics through Logic-Based Modelling and Simulation

Analysing Supply Chain Operation Dynamics through Logic-Based Modelling and Simulation

Areti Manataki

Doctor of Philosophy

Centre for Intelligent Systems and their Applications

School of Informatics

University of Edinburgh

2012

Abstract

Supply Chain Management (SCM) is becoming increasingly important in the modern business world. In order to effectively manage and integrate a supply chain (SC), a deep understanding of overall SC operation dynamics is needed. This involves understanding how the decisions, actions and interactions between SC members affect each other, and how these relate to SC performance and SC disruptions. Achieving such an understanding is not an easy task, given the complex and dynamic nature of supply chains. Existing simulation approaches do not provide an explanation of simulation results, while related work on SC disruption analysis studies SC disruptions separately from SC operation and performance.

This thesis presents a logic-based approach for modelling, simulating and explaining SC operation that fills these gaps. SC members are modelled as logic-based intelligent agents consisting of a reasoning layer, represented through business rules, a process layer, represented through business processes and a communication layer, represented through communicative actions. The SC operation model is declaratively formalised, and a rule-based specification is provided for the execution semantics of the formal model, thus driving the simulation of SC operation. The choice of a logic-based approach enables the automated generation of explanations about simulated behaviours. SC disruptions are included in the SC operation model, and a causal model is defined, capturing relationships between different types of SC disruptions and low SC performance. This way, explanations can be generated on causal relationships between occurred SC disruptions and low SC performance.

This approach was analytically and empirically evaluated with the participation of SCM and business experts. The results indicate the following: Firstly, the approach is useful, as it allows for higher efficiency, correctness and certainty about explanations of SC operation compared to the case of no automated explanation support. Secondly, it improves the understanding of the domain for non-SCM experts with respect to their correctness and efficiency; the correctness improvement is significantly higher compared to the case of no prior explanation system use, without loss of efficiency. Thirdly, the logic-based approach allows for maintainability and reusability with respect to the specification of SC operation input models, the developed simulation system and the developed explanation system.

Acknowledgements

These years have been an exciting and challenging adventure. I was lucky to be accompanied by wonderful people. As my grandmother says, good company gets joy multiplied and sadness divided. So here we go:

I would like to thank my supervisors, Jessica Chen-Burger and Michael Rovatsos, for their guidance and support throughout the PhD process. Jessica, the long hours of discussion and brainstorming spent together managed to motivate me, even at moments when I was losing faith. Michael, your input was invaluable and I have learnt a great deal from you.

A big thank you to my examiners, Alan Smaill and Henk Akkermans, for the incredibly enjoyable viva. Your feedback was to the point, and your comments helped improve the content of this thesis.

I would also like to thank all the proof readers of this thesis, for their thorough and helpful advice. You know who you are.

Friends and colleagues have brightened up my time here. My special thanks to Alexandros-Sotiris Belesiotis and George Christelis, who have supported me and inspired me all along. Thanks to Tommy French, Lysimachos Zografos, Matt Crosby and Francesco Figari for the good times spent together. And thanks to my officemates for the never-ending coffee breaks.

I am deeply grateful to my parents, Manolis and Katerina, and my sister, Yianna. Your encouraging and stimulating attitude has made a great difference to this journey. I could not have wished for more patient and understanding support.

Finally, my thanks to Dimitris. For everything.

I will always remember this journey with a smile. Thank you all!

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Early work was presented in the following conferences:

- Manataki, A., Chen-Burger, Y.-H., and Rovatsos, M. (2010) Improving the understanding of supply chain dynamics: Towards an intelligent simulation tool. In *Proceedings of the 17th Conference of the European Operations Management Association (EurOMA 2010)*, Porto, Portugal.
- Manataki, A., Chen-Burger, Y.-H., and Rovatsos, M. (2010) Towards Improving Supply Chain Coordination through Agent-Based Simulation. In *Proceedings of the 8th International Conference on Practical Applications of Agents and Multiagent Systems (PAAMS 2010)*, Salamanca, Spain.

(Areti Manataki)

Table of Contents

Chapter 1

Introduction

The modern business landscape is highly dynamic and competitive. Firms are faced with the challenge of meeting changing customer requirements while keeping costs low so as to survive in the global marketplace. In this context, companies can no longer compete in isolation from their Supply Chain (SC) partners. It has been acknowledged that the business world has now entered an era of SC- rather than enterprise-based competition (Harrison and van Hoek, 2008), and thus "supply chain management consciousness is accelerating up the corporate agenda" (Storey et al. 2006, p.757).

A systemic view of Supply Chain Management (SCM) is becoming prominent, and there is a need for SC integration. Achieving coordinated and fully integrated supply chains is a challenging task that requires a deep understanding of SCM dynamics. The SCM research community has long recognised the significance of analysing SCM dynamics. There is an extensive body of work on SCM dynamics in the context of SC planning and demand forecasting (Lee et al. 1997; Riddalls et al. 2000; Hwarng and Xie, 2008). Studies have also appeared on the matter of SC configuration considering aspects of SCM dynamics (Akkermans, 2001; Choi et al. 2001). However, the problem of analysing SC operation dynamics remains understudied.

SC operation dynamics involve the interrelations and the impact of SCM decisions and activities of individual SC members on other SC members and the supply chain as a whole. They also involve the relationships between disruptive events that occur during SC operation, a phenomenon that is not uncommon. In order

to understand SC operation dynamics, a mechanism for capturing and explaining such causal relationships is needed. Understanding SC operation dynamics is an important problem, as it is a prerequisite for supply chain integration. It is also a hard problem, given the distributed, dynamic and complex nature of modern supply chains.

Knowledge-based techniques (Stefik, 1995) are useful for analysing complex and dynamic systems. They provide transparent and rigorous reasoning mechanisms that allow the capturing and explanation of complex behaviours. They also enable the diagnosis of problematic situations, which can be supported by valuable explanations.

This thesis proposes a knowledge-based approach for analysing supply chain operation dynamics. SC operation is modelled in a declarative fashion and it is simulated following rule-based execution semantics. This approach facilitates the explanation of simulated SC operational behaviours and performance, and it allows for diagnosing problematic SC operation.

1.1 Research Problem

Consider a supply chain consisting of several parties that contribute to the delivery of products to the final customers. During SC operation, the members of the supply chain perform activities giving rise to a flow of products, funds and information across the supply chain. They make decisions, act and interact with each other in order to fulfil final customer requests. At this operational level, planning and configuration issues are considered already defined, while SC members perform procurement, manufacturing and replenishment operations. There are interdependencies between these operations that are internal to each SC member, resulting in complex inter-functional dynamics. There are also interdependencies between the decisions, actions and interactions of different SC members, understood as inter-organisational dynamics. Overall SC performance depends heavily on the successful management of these two types of interdependencies.

Disruptive events tend to occur during SC operation, such as delays and machine breakdowns. These can be caused by and have a direct effect on the decisions,

actions and interactions of different SC members. SC disruptions can propagate across the supply chain, give rise to new types of SC disruptions and lead to low SC performance. These SC disruption-related interdependencies are another facet of SC operation dynamics.

Before defining the research problem, let us mention that we shall use the term 'operational behaviour' to refer to the decisions, actions and interactions of an SC member during SC operation. The problem of analysing SC operation dynamics involves identifying the interrelationships that lie:

- between different aspects of an SC member's operational behaviour
- between the operational behaviour of different SC members
- between SC members' operational behaviour and SC performance
- between SC members' operational behaviour and SC disruptions
- between SC disruptions across the supply chain
- between SC disruptions and SC performance

As it will be discussed in Chapter 3, existing work does not address all these points in parallel. Our aim is, thus, to support a joint study of these issues.

1.1.1 Motivating Example

We will now introduce an example to illustrate and motivate the problem of analysing SC operation dynamics. This example will be used throughout the thesis to demonstrate aspects of our approach. Consider the supply chain depicted in Figure 1.1, consisting of eight SC members. These firms interact, make decisions and perform activities that support the flow of several types of products along this supply chain. There are direct and indirect dependencies between these SC members, and several aspects of their SC operational behaviour are interrelated. Furthermore, different disruptive events occur during SC operation, and SC performance is low.

Analysing the SC operation dynamics for this supply chain involves answering questions such as: Why does Supplier2 begin a production activity at timepoint 4? Why is the cost at Supplier3 1330? What are the root causes of Manufacturer's low on time rate? Why do the products required for delivering at Supplier4 become

available late? What are the effects of the damage of products at Supplier1 on the entire supply chain and SC performance? Does the delivery delay at Supplier1 cause the making delay at Supplier4? The answers to such questions are valuable, as they can guide SC improvement and enhance the coordination of activities across the supply chain. Identifying the answers to such questions is not an easy task, as supply chains are complex systems that consist of members with rich and dynamic operational behaviour. Therefore, an intelligent solution is required.

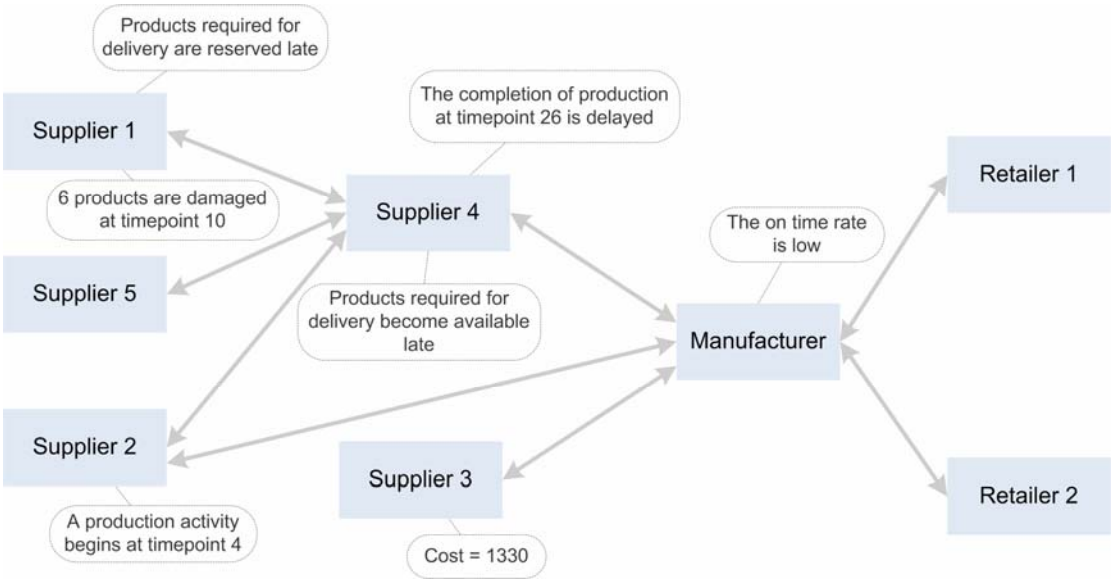

Figure 1.1: A supply chain with complex SC operation dynamics.

1.2 A Logic-Based Approach

In order to provide a practical solution to the problem of analysing SC operation dynamics, three points need to be addressed: (1) The solution should be tailored to the SC operation domain, so that it can be easily used by SCM practitioners. (2) It should support the end-to-end analysis of complex supply chains and facilitate the experimentation with different SC operation scenarios. (3) It should provide automated support, so that questions like the ones presented in the previous section can be directly and explicitly answered.

We address the first point by conceptualising SC operation based on appropriate SCM theory and standards. We identify constructs that cover commonly agreed aspects of SC operation, while recognising current trends and issues of the field.

These constructs are declaratively specified and they can be directly used to model SC operation scenarios.

We recognise simulation as a useful method for addressing the second point, as it can provide an insight into the operation of complex and dynamic systems, and facilitate what-if analysis. Therefore, we develop a transparent rule-based executable model of SC operation and we implement a simulation environment that supports instances of this model. The developed system can be used to simulate SC operation, measure SC performance and detect SC disruptions.

The adoption of a logic-based approach contributes towards addressing the third point. Utilising the declarative formalism of SC operation constructs and the rule-based specification of execution semantics, we design a mechanism for generating explanations of simulated SC operation. An explanation system is implemented which can automatically answer questions on SC operation dynamics, such as the ones mentioned for the supply chain of Figure 1.1.

We name this thorough and practical solution to the problem of analysing SC operation dynamics 'SCOlog' (Supply Chain Operation dynamics explained through a LOGic-based approach). SCOlog employs a logic-based approach to analysing supply chain operation dynamics. It consists of a formal model of SC operation that is tailored to the domain, an executable model of SC operation and a mechanism for generating explanations of simulated SC operation. The value of SCOlog is demonstrated through the use of appropriately implemented simulation and explanation systems.

1.2.1 Research Statement

The hypothesis underlying the research presented in this thesis is as follows:

> *SCOlog generates explanations which provide useful insight into supply chain operation dynamics and employs a logic-based approach to the modelling and simulation of supply chain operation, allowing for maintainability and reusability.*

Here, by maintainability it is meant that one can modify the resulting model without much effort. By reusability we mean that components of the resulting model can be

used in a different context. Usefulness refers to two aspects: the performance of SCM experts when explaining SC operation dynamics, and the understanding of the domain for non-SCM experts, that result from the use of the tool.

The thesis claims are as follows:

1. Automated explanation support is useful for the task of explaining supply chain operation dynamics, allowing for users' higher (a) time-efficiency, (b) correctness and (c) certainty about the explanations provided compared to the case where such support is not available.
2. The use of automated explanation support improves the performance of non-SCM experts, with respect to their (a) time-efficiency and (b) correctness when explaining SC operation dynamics. The correctness improvement is bigger compared to the case where no automated explanation support is available, without loss of time-efficiency. This suggests that the use of automated explanation support improves the understanding of the domain for non-SCM experts.
3. A logic-based approach for modelling, simulating and explaining SC operation scenarios allows for maintainability and reusability with respect to (a) the specified SC operation input models, (b) the developed simulation system and (c) the developed explanation system.

1.2.2 Research Contributions

The main contributions of this research are outlined as follows:

- **Formal model of SC operation:** We conceptualise SC operation by identifying structural, behavioural and disruption-related constructs. The conceptual model is formalised in a declarative fashion and with the use of advanced IT-inspired technical abstractions, such as intelligent agents, business processes and business rules. Furthermore, we specify a causal model of problematic SC operation, defining causal relationships between different types of SC disruptions and low SC performance. The resulting formal model is tailored to the SCM domain and is shown to be maintainable and reusable.

- **Executable model of SC operation and implemented simulation system:** We specify the execution semantics of the formal model in a rule-based manner, based on which dynamic behaviours can be driven. Given these execution semantics, we provide an algorithm for simulating system-wide SC operation. An appropriate simulation system is implemented, consisting of modules such as a workflow engine and an SC disruption detector. This simulation environment can be used to analyse and experiment with different SC operation scenarios, and it is shown to be easy to maintain and reuse.
- **Mechanism for generating explanations of SC operation and implemented explanation system:** Given the rule-based specification of the execution semantics, we describe a framework for explaining dynamic SC operational behaviours. We also provide a mechanism for diagnosing and analysing problematic SC operation based on the specified causal model. An appropriate explanation system is implemented, which automatically generates explanations about SC operation dynamics at two levels of detail. This automated explanation support is shown to be useful to both SCM experts and non-experts. Furthermore, the explanation environment is characterised by maintainability and reusability.

We identify the following research areas that benefit from the work presented in this thesis:

- **Supply Chain Management:** A thorough and tested framework is provided for analysing SC operation dynamics, a problem that is currently understudied. The specified formal model of SC operation can also be of use in the area of SC modelling. We contribute to the area of SC disruption analysis through a powerful method for analysing problematic SC operation; this method is tailored to the SCM domain and it is found to be maintainable. Finally, the implemented simulation and explanation systems provide a practical contribution to the field.
- **Simulation:** A generic executable model and a simulation algorithm are provided, which can be used for simulating systems consisting of several members that think, act and interact. Another contribution is the generic

mechanism for generating explanations of simulation results. Furthermore, the implemented simulation and explanation systems can serve as solutions against which different simulation approaches can be evaluated.

- **Knowledge-Based Systems:** We develop a maintainable mechanism for generating explanations of the dynamic behaviour of complex systems. This mechanism is based on the combination of technologies such as workflows, intelligent agents and reasoning engines. We also provide an explanation system that is considered to be useful by users.

1.3 Thesis Structure

The research presented in this thesis lies at the intersection of Supply Chain Management, Knowledge-Based Systems, Workflow Management Systems and Multiagent Systems. An overview of these areas is provided in Chapter 2, which serves as background knowledge for this thesis. Chapter 3 discusses work that is relevant to the problem of analysing SC operation dynamics. SC simulation and SC disruption analysis approaches are presented, and research gaps are identified.

The literature review chapters are followed by two chapters on the research methods. Chapter 4 presents the modelling framework proposed in this thesis. SC operation is conceptualised through appropriate constructs and formalised in a declarative fashion. Chapter 5 details the approach to simulating and explaining SC operation, which is driven by the rule-based specification of the execution semantics of the formal model. Illustrative examples are provided to show the value of automated explanation.

Chapter 6 provides a detailed evaluation of SCOlog with respect to the research claims. We empirically evaluate the usefulness of the approach for explaining SC operation dynamics and improving the understanding of the domain for non-SCM experts. We also discuss aspects of maintainability and reusability. In the final chapter of the thesis we conclude and identify a number of avenues for future research.

Chapter 2

Background

The research project discussed in this thesis is of a multidisciplinary nature, as it attempts to provide a solution to a business problem (i.e. the analysis of supply chain operation dynamics) with the use of Artificial Intelligence techniques. This chapter is a broad and shallow overview of research areas that are loosely related to this project, thus providing the background needed for understanding the context and content of this thesis. Alternative approaches to the research problem are not presented here – Chapter 3 is dedicated to the in depth discussion of related work. The following areas are discussed in this chapter: Supply Chain Management (Section 2.1), Business Process Modelling and Workflow Management Systems (Section 2.2), Intelligent Agents and Multiagent Systems (Section 2.3) and Knowledge-Based Systems (Section 2.4). Basic concepts and trends are explained, and the relevance of each theme to this project is highlighted.

2.1 Supply Chain Management

A *Supply Chain* "consists of all parties involved, directly or indirectly, in fulfilling a customer request" (Chopra and Meindl 2003, p.4). Figure 2.1 presents a typical supply chain, as well as the involved flows of products (downstream), funds (upstream) and information (across the supply chain). It is worth mentioning that an organisation can be a member of several supply chains, depending on the product. There is a wide range of supply chains in the business world, from simple to very

complex ones. A simple supply chain may consist of a few suppliers, a manufacturer and a customer; a complex supply chain can include several tiers of suppliers of raw materials and components, a manufacturer, numerous wholesalers, retailers and customers, and several tiers of distributors and warehouses. Figure 2.2 presents a complex SC network structure, consisting of multiple tiers across the supply chain, and several members within each tier.

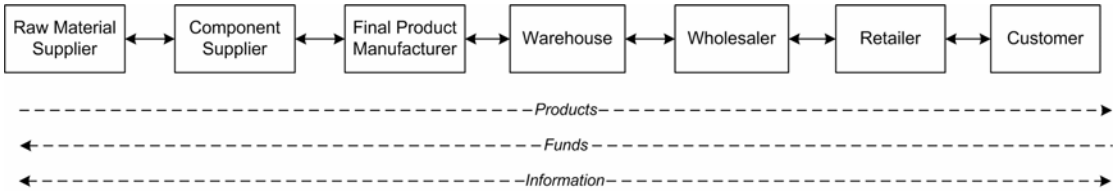

Figure 2.1: A typical supply chain

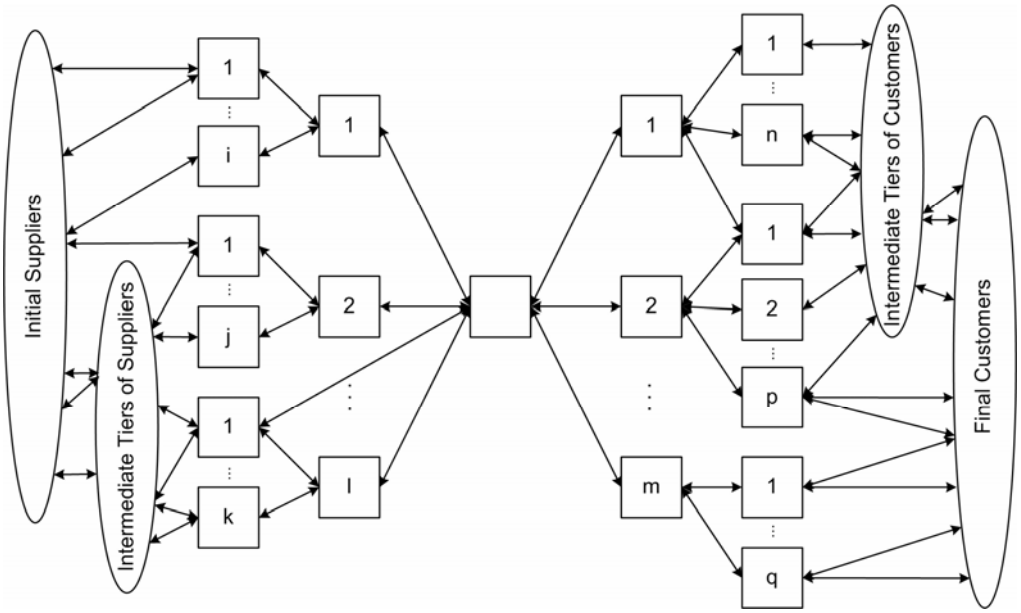

Figure 2.2: A complex supply chain network structure
(adapted from Lambert and Cooper(2000))

Supply Chain Management "involves the management of flows between and among members of the supply chain in order to maximise total supply chain profitability" (Chopra and Meindl, 2003, p.6). This is not an easy task, especially given the complex and dynamic nature of supply chains. The behaviour of individual SC members affects the decisions and activities of other SC members, thus influencing their performance. Hence, overall SC performance depends on the strategies, policies and actions of each participating organisation. However, maximising the local performance of participating companies does not guarantee the maximisation of the

performance of the entire SC network; on the contrary, it may lower overall SC performance. Self-optimising, opportunistic behaviour of individual SC members is in conflict with the objectives of SCM (Tan, 2001). This fact makes the task of supply chain management particularly hard.

Choosing appropriate *SC performance measures* is crucial in order to successfully analyse and enhance overall system performance. We can identify three SC performance measurement frameworks as the most prominent in the field. First, the framework proposed by Beamon (1999) identifies metrics along three categories of performance measures: resource, output and flexibility. Resource-related metrics involve the efficiency of resource usage, and they include total cost and return on investment. Output-related metrics involve customer service, and they include on time deliveries, quality and fill rates. Flexibility-related metrics involve how well the supply chain reacts to uncertainty, and they include volume and delivery flexibility. Second, Gunasekaran et al. (2001) present metrics for performance along two dimensions: SC links (i.e. planning, sourcing, production, delivery and customer service and satisfaction) and SC levels (i.e. strategic, tactical and operational). Third, the Supply Chain Operations Reference (SCOR) model (Supply Chain Council, 2008) includes a hierarchical framework of SC metrics along five performance attributes: reliability, responsiveness, agility, costs and assets. Examples of SCOR-based metrics for each category include: delivery performance to customer commit date, source cycle time, upside deliver flexibility, cost to make and deliver fixed asset value. In this work we adopt the SCOR-based framework for SC performance measurement for three reasons. First, it is an extensive framework, providing metrics for the main aspects of SC performance. Second, given its hierarchical structure, it is easy to use in the context of large supply chains with complex operations. Third, it explicitly specifies the calculation for each metric, while linking it to involved processes. We recognise that the SCOR model suffers from a considerable drawback, i.e. the fact that quality aspects are considered only in a limited way. Product quality is important in the context of SCM, as it is highly related to customer satisfaction. Nevertheless, this is not a limitation for this work, as we focus on quantitative rather than qualitative performance metrics, which are easier to measure in an SC operational setting. Let us now provide the definitions of two SCOR-based metrics

that will be mentioned in chapters 4 and 5. On time rate, which corresponds to SCOR's metric 'delivery performance to customer commit date' is the percentage of orders that are delivered on the time promised. Cycle time is the average time associated with some SCM operation (e.g. make cycle time is the average time associated with production). Note that the SCOR model is also discussed in Section 2.1.4, where its process-related aspects are presented.

Managing a supply chain requires the management of processes within and across organisational borders, such as customer relationship management, customer service management, demand management, order fulfilment, manufacturing flow management, procurement, product development and commercialisation, and returns (Lambert and Cooper, 2000). Given this breadth of scope, the field of SCM is characterised as interdisciplinary (Burgess et al. 2006). Areas such as marketing, logistics, purchasing, operations management and strategy are closely related to supply chain management.

Event though there has been active research in SCM since the 1990s, the field is still developing and far from mature. There is little consensus on the definition and the scope of SCM, very often causing confusion. Nevertheless, SCM is typically understood in terms of processes (a chain of activities) or systems (interrelated processes, concepts, networks and frameworks), and the two most broadly used constructs for SCM are process improvement orientation and inter-organisational relationships (Burgess et al. 2006).

2.1.1 Trends in SCM

A current trend in SCM is a shift from the antagonistic model to a collaborative model (Storey et al. 2006), thus there is a requirement for the full alignment and integration of supply chains. A systemic and *holistic* view of SCM is becoming prominent, where the supply chain is viewed as a "virtual organization composed of several independent entities with the common goal of efficiently and effectively managing all its entities and operations, including the integration of purchasing, demand management, new product design and development, and manufacturing planning and control" (Tan, 2001). Thus, the supply chain should have a common mission, goals and objectives as a whole, but at the same time individual SC

members can pursue their independent policies. Holistic SCM is in line with the observations on current competition that "one of the most significant paradigm shifts of modern business management is that individual businesses no longer compete as solely autonomous entities but rather as supply chains" (Lambert and Cooper, 2000, p.65) and that "firms are finding that they can no longer compete effectively in isolation of their suppliers or other entities in the supply chain" (Lummus and Vokurka, 1999, p.11). Although the SCM research community promotes a holistic view of the field, most of the existing research still focuses on specific SC links or nodes, as pointed out by Giunipero et al. (2008). The authors conducted an SCM literature review and recognised a necessity for future studies to expand their focus beyond one-tier supplier-buyer relationships. Moreover, the practice of SCM is still far from the vision of SC integration, as supply chains often fail to behave as one entity (Holweg and Pil, 2008).

Lean SCM is a paradigm widely discussed by both SCM scholars and practitioners. Lean thinking refers to the elimination of waste and a focus on value (Hines et al. 2004). Lean SCs remove any non-value adding activities along the entire SC network and focus on the core aspects of the overall value chain. A shift towards a systemic view of lean SCM has been identified by Hines et al (2004), thus following the wider trend of holistic SCM.

Another trend in SCM is *agility*, which is "the ability of an organisation to respond rapidly to changes in demand" (Christopher 2000, p.38). Essentially, agile SCs are flexible and responsive to unexpected changes in supply or demand (Lee, 2004). Agility is an important SC capability given the current business landscape, characterised by globalisation, rapid rhythms of change and high degree of uncertainty. It is interesting to note that contributing factors towards SC agility, such as process, network and virtual integration, as identified by Christopher (2000), are closely related to the holistic view of SCM.

2.1.2 SCM Problems and Dynamics

Three main categories of *SCM problems* can be identified: (1) SC planning and demand forecasting, (2) SC configuration and (3) SC operation. SC planning and demand forecasting is the problem of estimating future demand across the different

SC tiers, thus feeding the manufacturing plan of each SC member. One of the most important problems within this category is the bullwhip effect, which refers to the amplification of demand order variability as we move up in the supply chain (Lee et al. 1997). SC configuration involves the specification of the SC system's structure, policies and processes in a static way. Selecting suppliers, identifying the location of facilities and choosing information exchange mechanisms are problems that fall into this category (Chandra and Grabis, 2007). SC operation refers to the actions and interactions between SC members, leading to the flow of materials, funds and information across the supply chain. It is worth mentioning that these three problem categories are interdependent: SC planning decisions affect SC configuration aspects, which, in turn, have an impact on the daily SC operation. They are all important, as incorrect SC planning and demand forecasting would lead to a sub-optimal SC configuration, resulting in operational problems along the SC and low SC performance.

When solving SCM problems, one needs to consider the complex interrelations between SC members and the dynamics involved. For example, forecasting future demand at some SC node is heavily influenced by demand forecasting practices and replenishment policies at subsequent tiers. Similarly, the location of facilities for the suppliers of some organisation affects the decision on the location of its facilities. As far as SC operation is concerned, external processes, such as suppliers' or customers' processes, impact an organisation's internal processes, but these are often neglected (Barratt, 2004). Given that this latter problem is understudied, yet important, the focus of this thesis is on SC operation dynamics.

Coordinating activities across the supply chain requires a deep and solid understanding of SC operation dynamics, and is a prerequisite for SC integration. According to Lee and Whang (2004), workflow coordination is one of the four dimensions of SC integration, while Simatupang et al. (2002) recognise logistics synchronisation as one of the four modes of coordination that affect operational performance and SC integration. This illustrates the significance of understanding SC operation dynamics, and motivates our research.

2.1.3 SC Disruptions

The *management of SC disruptions* across the supply chain is a problem closely related to SC operation dynamics. SC disruptions are "unplanned and unanticipated events that disrupt the normal flow of goods and materials within a supply chain" (Craighead et al. 2007). These events may occur at the organisation level (e.g. machine breakdown, damage of products), the supply chain level (e.g. delay or unavailability of materials, demand discrepancies, transportation delays) or the wider environment level (e.g. natural disasters, terrorist attacks, wars, economical crises). SC disruptions can have severe effects on the SC system, typically expressed in terms of high costs and low responsiveness. The occurrence of disruptive events along the supply chain and the resulting poor performance are becoming more and more common, mainly due to SC globalisation and the increased use of outsourcing practices. The growing complexity of supply chains makes it difficult to manage SC risk and disruptions (Butner, 2010). It follows that SC disruptions and SC risk are perceived as one of the most important current and future issues of the field by SCM practitioners (Butner, 2010; Melnyk et al. 2009). Similarly, the study of minimising SC disruptions and uncertainties is regarded as a fruitful research opportunity (Stock et al. 2010).

In order to effectively mitigate SC risk and manage SC disruptions, one needs to understand how SC disruptions affect SC operation at a local and a global level. This involves understanding how disruptions may propagate across the supply chain, and what impact they may have on individual and overall SC performance. It has been argued that "there has been relatively little reported in the important area of understanding the system-wide or global impact of SC disruptions both upstream, downstream and laterally in the SC system" (Blackhurst et al. 2005, p.4076). For this reason, in our research we specifically consider SC disruptions when exploring SC operation dynamics.

2.1.4 SC Modelling

Supply Chain Modelling assists the analysis of complex supply chains, thus facilitating SCM and supporting SC integration. SC modelling is recognised as a prerequisite for SC integration (Min and Zhou, 2002; Li et al. 2002). According to Li

et al. (2002), the main motivations for SC modelling are the following: (1) to capture SC complexities by representing the SC in a uniform way, (2) to design and specify SCM processes across the entire SC network, (3) to communicate and agree on the vision to be shared by SC partners and (4) to reduce SC dynamics during the SC design phase.

A *taxonomy* of SC models is provided by Beamon (1998) and Min and Zhou (2002), consisting of deterministic analytical, stochastic analytical, economic models and simulation models in the first case, and deterministic, stochastic, hybrid and IT-driven models in the latter case. It has been argued that analytical models are often too simplistic to deal with highly complex supply chains, while simulation models allow for a more "realistic optimisation" in such a case (Hung et al. 2006). As far as SC modelling approaches are concerned, we distinguish the operational and agent-oriented approach, thus covering the two main aspects of SCM, i.e. process improvement orientation and inter-organisational relationships.

Two of the most popular models for SCM, which adopt the operational approach, are the Supply Chain Operations Reference (SCOR) model (Supply Chain Council, 2008) and the Global Supply Chain Forum (GSCF) framework (Lambert and Cooper, 2000; Lambert, 2008). The *SCOR model* is developed by the Supply Chain Council and is a process reference model for the whole SC, thus viewing the SC as a chain of processes. It is perceived as a strategic planning tool, particularly useful for top managers, and is regarded as a standard by the SCM community (Bolstorff and Rosenbaum, 2012). It consists of standard descriptions of SC processes, a framework of relationships between them, standard metrics for performance measurement and best practices in the field. It is a hierarchical model, consisting of three levels of processes, as shown in Figure 2.3:

1. **Top level:** Five distinct management processes are identified at this level, i.e. plan, source, make, deliver and return. Planning involves identifying a course of action in order to balance supply and demand, and achieve SC objectives. Sourcing involves the procurement of goods or services to meet planned or actual demand. Making covers production activities, adding value to or transforming components to finished goods. Delivering provides

finished goods or services to fulfil customer orders. Returns are associated with returning or receiving returned products for any reason.

2. **Configuration level:** Processes at this level are classified in two different ways. Firstly, three process types are identified: planning, execution and enabling. Planning processes align expected resources to meet anticipated demand requirements over a specified planning horizon. Execution processes make use of resources to change the business state with respect to availability of products. Enabling processes support planning and execution processes through the preparation, maintenance and management of needed information or relationships. The second classification of processes at this level is based on the adopted manufacturing strategy, thus distinguishing between make-to-stock, make-to-order and engineer-to-order processes.

3. **Process element level:** Processes at this level are a decomposition of configuration level processes. Additional information is provided for each decomposed process, such as inputs and outputs, related performance attributes and best practices.

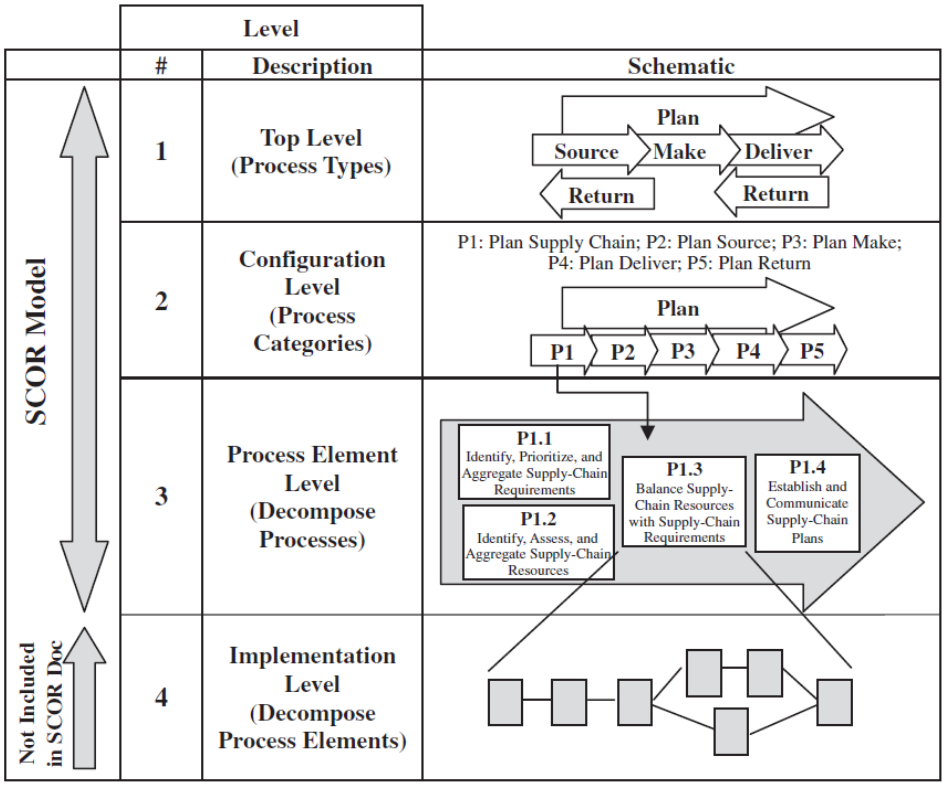

Figure 2.3: Levels of processes at the SCOR model

The *Global Supply Chain Forum framework* views SCM as the integration of the following eight key business processes along the SC (Lambert and Cooper, 2000; Lambert, 2008): (1) customer relationship management, (2) customer service management, (3) demand management, (4) order fulfilment, (5) manufacturing flow management, (6) supplier relationship management, (7) product development and commercialisation and (8) returns management. Each of these processes is cross-functional and cross-firm. The GSCF framework defines strategic and operational sub-processes for each of these key business processes, as well as any interfaces between them.

A comparison between the SCOR model and the GSCF framework has been conducted by Lambert et al. (2005) across several criteria, such as scope, intra- and inter-company connectedness and drivers of value generation. It was found that the GSCF framework has a wider scope and is more strategic, focusing on cross-functional and cross-firm connectedness. On the other hand, the SCOR model has a more operational orientation, linking explicitly processes and performance metrics, and it is easier to implement. In this work we adopt the SCOR model given its operational orientation and coverage of both SCM processes and performance metrics.

2.1.5 Desired Properties of a Solution

Given the state of the art in SCM, as discussed in this section, we identify the following desired properties of a solution to the studied research problem:

1. **Holistic view:** Recognising the trend of holistic SCM on one hand, and the limited research on SC-wide behaviours on the other hand, there is a need to study system-wide SC operation and overall SC performance.
2. **Include SC disruptions:** Given the close relationship between SC disruptions and SC operation dynamics, there is a need for a joint study of these two issues.
3. **Cover standard aspects of SC operation:** SCM is characterised by large breadth of scope. Therefore, it is important to cover the main aspects of SC operation, as well as key SC disruption types.

4. **Deal with complex situations:** Supply chains can have a complex structure, and SC operation decisions and behaviours can be complex. When studying SC operation dynamics, one should consider such complexity issues.
5. **Include flexibility aspects:** Given the trend of SC agility, there is a need to consider flexibility decisions and behaviours when studying SC operation dynamics.
6. **Facilitate what-if analysis:** Supply chains are considered to be dynamic systems. Therefore, a solution to the studied problem should facilitate the experimentation with different SC configurations. This also requires that modifying the specification of different SC configurations should not require much effort.
7. **Maintainability:** It should be easy to update a solution to the problem to incorporate any theoretical advances, especially since SCM is a field still under development.

2.1.6 Relevance to the Project

This project focuses on supply chain management, and more specifically SC operation. The discussed SCM trends were taken into account when designing a solution approach to the research problem. For instance, recognising the importance of SC agility, we incorporate decision-making for flexibility purposes, which from now on we will call 'flexibility decision-making'. Furthermore, we consider SC performance measures when modelling the domain; particularly, we use SCOR-based metrics. The SCOR model is also used for modelling the operations of SC members in our approach. Finally, we regard SC disruptions as an important aspect of SC operation dynamics, and thus we explicitly model, simulate and analyse their occurrence.

2.2 Business Process Modelling and Workflow Management Systems

Business Process Modelling (BPM) is a widely-used approach that "allows the capturing, externalisation, formalisation and structuring of knowledge about enterprise processes" (Kalpic and Bernus, 2002). A business process is defined as "a structured, measured set of activities designed to produce a specified output for a particular customer or market. Implying a strong emphasis on how work is done, it is a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs" (Davenport, 1993). There are several advantages of business process modelling. Firstly, BPM techniques capture informal and abstract activities within an enterprise and make them concrete, thus allowing better understanding and communication of business operations. Secondly, formal analysis of business processes (e.g. through simulation) can assist performance measurement and therefore guide process improvement; this was the main driver behind the Business Process Reengineering (BPR) wave of the 1990s. Thirdly, business process models can support the design of information systems as well as software development.

Workflow is closely related to BPM and BPR, as it is concerned with the automation of procedures within an organisation (Georgakopoulos et al. 1995). Management of a workflow involves process modelling and workflow specification, process reengineering, and workflow implementation and automation (Georgakopoulos et al. 1995). A Workflow Management System (WfMS) is a system that "completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic" (Hollingsworth, 1994, p.6). According to Mentzas et al. (2001) there are three basic categories of workflow techniques: communication-based, focusing on commitments among humans, activity-based, focusing on the work, and hybrid techniques. Workflow technologies can be useful in two main ways. Firstly, the process execution and enactment support that they offer can automate organisational procedures, thus reducing costs and increasing efficiency. Secondly,

workflow-based simulation can assist the analysis and the improvement of business operations.

Popular business process modelling techniques and languages include UML's Activity Diagrams (OMG, 2004) and Integration DEFinition language (IDEF3) (Mayer et al 1995), as well as the more recent Business Process Model and Notation (BPMN) (OMG, 2011), Web Services Business Process Execution Language (WSBPEL) (OASIS, 2007) and Process Specification Language (PSL) (Grüninger and Menzel, 2003). As far as workflow representation is concerned, Petri Nets are a widely used notation (van der Aalst, 1998), while YAWL (van der Aalst and ter Hofstede, 2005) is a more recent language. It is also worth mentioning the Workflow Reference Model (Hollingsworth, 1994), a standard suggested by the Workflow Management Coalition, which identifies the characteristics, functions and interfaces of workflow systems.

A language that supports business process modelling and workflow system development is the Fundamental Business Process Modelling Language (FBPML) (Chen-Burger et al. 2002). It is a merger of PSL and IDEF3 and provides both formal semantics and rich visual modelling methods. These attributes make it useful for workflow execution and analysis. FBPML is an activity-based language, which is role-aware and that contains communication elements. A model specified in FBPML consists of main nodes, junctions, links and annotations. Main nodes represent processes (i.e. activities or tasks) within a model, and their declarative definition includes the specification of triggers, preconditions and actions. Links and junctions specify the control flow of processes within a business process model, thus allowing sequencing and branching for selection or parallelisation purposes. The four types of junctions available in FBPML (i.e. start, finish, and, or) can be combined to define complex structures. And- and or-junctions can be used in a split or joint context, and their semantics are formally defined. Figure 2.4 presents an illustrative business process model in the FBPML notation.

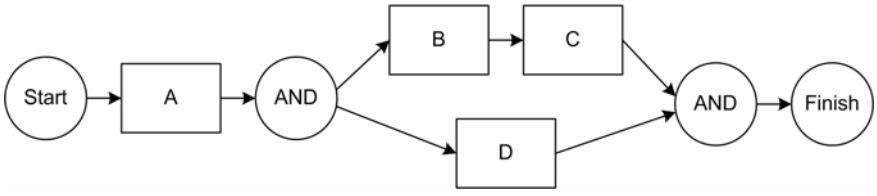

Figure 2.4: Example of a business process model in FBPML

Business process modelling and workflow management are regarded as useful methodologies for SCM, especially since SCM is widely understood in terms of processes. The advantages of business process modelling discussed earlier still hold in a SCM context. It is, thus, not surprising that the SCOR model and the GSCF framework adopt a process-oriented view of SCM. Research that applies BPM techniques on SCM includes Trkman et al. (2007) and Wang et al. (2010). Trkman et al. (2007) utilise business process modelling and simulation for reengineering business processes across a supply chain, in an effort towards SC integration. Similarly, Wang et al. (2010) present a case study of BPR for a global supply chain; to this end, they employ a SCOR-based business process model as a basis for identifying problems in the supply chain.

Workflow technologies can be used in the context of SCM for automation or analysis purposes, and hence two streams of relevant research can be identified. The first stream of research deals with supporting e-SCs through workflow-enabled automation, an issue highlighted by Basu and Kumar (2002). Liu et al. (2005) present the architecture of an inter-enterprise workflow SCM information system, consisting of a WfMS at each SC member and an integrated interface. Goutsos and Karacapilidis (2004) utilise a workflow management module within an open SCM system that supports e-business transactions. Research in this area is still in its infancy, and it has been argued that "the occasions where WfMSs are consolidated with SCM software solutions in the same system or even in many integrated systems, are not proportionately numerous" (Tarantilis et al. 2008, p.1311). The second research stream employs workflow-based simulation to analyse SC operations. It is worth noting that several of the research papers discussed in Chapter 3 (i.e. on SC simulation and SC disruption analysis) adopt a workflow-based approach. Furthermore, Arns et al. (2002) propose a SC performance analysis framework based on Petri nets and queuing networks. Drzymalski and Odrey (2008) model SCOR-based processes with Petri nets and present a state-space method to calculate the remaining time for any order delivery.

2.2.1 Relevance to the Project

In this thesis we adopt a process-oriented approach for modelling the acting behaviour of SC members. FBPML is used for formalising SC business process models, as it has formal semantics and its declarative syntax supports workflow execution. The formalisation of business processes is presented in Section 4.3.2.2. We also develop a workflow engine in order to simulate the acting behaviour of SC members. Its main operations are discussed in Section 5.1.2.1.

2.3 Intelligent Agents and Multiagent Systems

An *intelligent agent* is defined as "a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives" (Wooldridge 2002, p.15). Intelligent agents have the following characteristics:

- autonomy: ability to operate without the direct intervention of humans or others, having control over their actions and internal state
- social ability: ability to interact with other agents in order to satisfy their design objectives
- reactivity: ability to perceive their environment, and respond to changes in a timely fashion in order to satisfy their design objectives
- pro-activeness: ability to exhibit goal-directed behaviour by taking the initiative in order to satisfy their design objectives

A *Multiagent System* (MAS) is a system of interacting intelligent agents. Agents within an MAS communicate in order to coordinate, whether they are competing or cooperating. In the case of cooperation, agents act in order to achieve a common goal, while pursuing their individual objectives.

Multiagent systems have been widely used within the SCM context, as they effectively match the nature of supply chains. As Moyaux et al. (2006, p.16) argue, "supply chains are made up of heterogeneous production subsystems gathered in vast dynamic and virtual coalitions; intelligent distributed systems, e.g. multiagent

systems, enable increased autonomy of each member in the SC". Each SC member can be represented by one or more agents with local goals and objectives, while the global MAS goals and objectives correspond to the holistic view of the supply chain. It is interesting to note that SC members have the same characteristics as agents (Moyaux et al. 2006):

- autonomy: The business operation of a company does not involve direct intervention of other companies, and each company has control over its actions and internal state.
- social ability: There is high degree of interaction between SC members.
- reactivity: Companies need to perceive their environment, especially their market and competition, and be flexible enough to respond to it in a timely and effective fashion.
- pro-activeness: Under the goal of maximising their profits, companies can take initiatives within their business operation.

Agent technologies are suitable for tackling all three main SCM problems, either by automating, simulating or recommending solutions. SC planning and demand forecasting can be facilitated though the advanced reasoning and communication capabilities of intelligent agents. Representative work in this area includes Fox et al. (2000), Liang and Huang (2006) and Zarandi et al. (2008). The work by Fox et al. (2000) is one of the earliest and most highly cited approaches for agent-oriented SCM, and it involves a framework for capturing coordination knowledge through conversation plans. The authors apply this framework in an SC planning setting, and show how demand forecasts are transformed into materials demand plans and production plans when communicated across different functions of an organisation. Liang and Huang (2006) present an agent-based approach for coordinating inventory across the SC and minimising overall SC cost. This coordination is achieved centrally through a genetic algorithm-enabled demand forecast agent, and under the assumption of full SC information sharing. A similar coordination mechanism is adopted by Zarandi et al. (2008), who show that the bullwhip effect can be reduced in a fuzzy environment. It is also worth mentioning the Trading Agent Competition for SCM (Collins et al. 2010), in which purely self-interested agents compete in a

setting with partial information. Each trading agent in the competition is responsible for sourcing components from suppliers, making different types of final products and selling them to customers. Hence, decisions on forecasting final demand and planning SC operations need to be made by trading agents.

SC configuration is enabled through the agents' learning and negotiation capabilities. MASCOT (Sadeh et al. 2001) is an agent-based architecture that supports the dynamic selection of SC partners within an open environment. Piramuthu (2005) provides an agent-based solution to dynamic SC formation and reconfiguration, in which a supplier is selected for each incoming order; this selection is based on the order attributes and is facilitated through machine learning techniques. The interaction for forming an SC network is studied by Fox et al. (2000) when applying a conversational coordination approach. In this work the choice of SC partners may involve negotiation with several rounds of proposals and counterproposals. Similarly, a coordinated iterative bidding mechanism is proposed by Akanle and Zhang (2008) for selecting a combination of suppliers for a customer order. The resulting networks for a set of orders are then clustered to recommend a future SC structure.

SC operation analysis can be facilitated though the reasoning and coordination capabilities of intelligent agents. Agent-based modelling is regarded to be useful for simulating SC operation, but an explicit study of SC operation dynamics is missing. Swaminathan et al. (1998), in one of the first research efforts in this area, suggest an agent-oriented modelling framework for SC simulation. They present an agent architecture and provide a library of structural and control elements to facilitate the specification of simulation models. Their work addresses all flows across the supply chain (i.e. products, information and funds), but it does not shed any light on how the activities of a single SC member affect other SC members. Allwood and Lee (2005) define an agent model for simulating and studying SC network dynamics with respect to demand amplification. In this model, each agent represents an SC member and consists of a strategic and an operational level, defined through appropriate mathematical functions. Even though basic operations are captured (e.g. order management, production planning and control, materials management and accounting), the dynamics between them and across the SC are not investigated.

Ivanov et al. (2010) adopt agent-based modelling within their framework for adaptive SC planning and operations control. This framework considers multiple SC aspects (e.g. product, functional, organisational, etc.) and includes SC operation simulation along these aspects; however, the focus is not on understanding the related operation dynamics, but on establishing effective adaptations for the SC configuration.

2.3.1 Relevance to the Project

Agent-based modelling is highly suitable for SCM, covering important characteristics of SC members' operational behaviour and providing a good abstraction of local and global SCM aspects. For this reason we employ intelligent agents for representing SC members and their thinking, acting and interacting behaviour. This is further discussed in Section 4.3.

2.4 Knowledge-Based Systems

A *knowledge-based system* (KBS) is "a computer system that represents and uses knowledge to carry out a task" (Stefik, 1995). KBSs are used to support human decision-making, learning and action. Similarly, expert systems are computer programs that use knowledge and inference mechanisms for solving problems that would normally require the knowledge of a human expert. They mimic the cognitive behaviour of a human expert when solving problems and making decisions (Giarratano and Riley, 1998). There are two main components of expert systems: a knowledge base and an inference engine. The knowledge base contains the knowledge needed for problem-solving in a particular domain; this often has a rule-based form. The inference engine draws conclusions by searching through and reasoning on the knowledge in the knowledge base. It is worth mentioning that sophisticated expert systems also include an explanation facility, which means that they can explain their reasoning for reaching some conclusion.

There are two central families of *inference algorithms*: forward and backward chaining. Forward chaining works forward from a set of known facts to conclusions that can be drawn based on them (Russel and Norvig, 2003). It is a form of data-

driven reasoning and it is the main idea behind production systems. Backward chaining works backward from a goal to a set of facts that support the goal (Russel and Norvig, 2003). It is a form of goal-directed reasoning, and it can support diagnostic tasks.

Logic programming is a computer programming paradigm that adopts a declarative approach (Russel and Norvig, 2003). This means that instead of encoding desired behaviours directly as program code, systems are constructed through the formal representation of knowledge. Prolog (Clocksin and Mellish, 2003) is a popular logic programming language that has been used in different contexts, including expert systems. Prolog programs are executed in a backward chaining fashion, while allowing for recursive search.

Knowledge-based systems can be used to assist or automate human problem solving in different ways. Firstly, they provide a means for the diagnosis of complex problems; when diagnosis is combined with an explanation facility, the reliability of KBSs increases considerably. This point is further discussed in Section 2.4.3. Secondly, they can tackle prognosis problems, mainly by devising forward-chaining knowledge inference. Thirdly, their transparency and explanation ability can be useful in an educational context, such as intelligent tutoring systems.

Knowledge-based systems can support or automate the decision-making on several SCM problems. Lawrynowicz (2007) tackles short-term production planning through the use of an expert system. As far as SC configuration is concerned, Vokurka et al. (1996) employ expert systems to support the selection of suppliers, while Isiklar et al. (2007) utilise rule-based and case-based reasoning for selecting third party logistics providers. The improvement of warehouse operations is studied by Chow et al. (2005), enabled through RFID-based real-time resource tracking and case-based reasoning support for resource management.

2.4.1 Business Rules

A *business rule* (BR) is "a statement that defines or constrains some aspect of the business" where the intention is to "assert business structure or control or influence the behaviour of the business" (Business Rules Group, 2000, p.4). Typically,

business rules describe business goals, problems, policies, regulations, etc. (Bajec and Krisper, 2005).

Four categories of BRs are identified by the Business Rules Group: (1) definitions of business terms, (2) facts relating terms to each other, (3) constraints and (4) derivations. The first two categories are structural assertions, as they describe aspects of the structure of the enterprise. The third category involves action assertions, as it imposes constraints on behaviour. The fourth category involves the derivation of knowledge given existing knowledge; this is similar to knowledge inference, discussed in the previous section. The classification provided by the Business Rules Group is broad, and there is a tendency in research and practice to use the term 'business rule' mostly for the last two categories. At the rest of this thesis, we will thus use the term 'business rule' for these two categories.

According to the Business Rule Approach (Ross, 2003), BRs should be made explicit and they should be expressed in a declarative fashion. Popular languages include the Object Constraint Language (OCL) (OMG, 2006) and the Semantics of Business Vocabulary and Business Rules (SBVR) (OMG, 2008), while general-purpose programming languages like Java and Prolog (Clocksin and Mellish, 2003) are still being used in the context of business rules. As far as software applications are concerned, we regard business rules engines and business rules management systems as the most prominent ones. Business rules engines are essentially inference engines that allow the execution of BRs; Jess (Friedman-Hill, 2003) is a widely used BR engine. Business rules management systems are more complete solutions, supporting the storing, execution, monitoring and maintenance of BRs; ILOG JRules (ILOG, 2005) and Drools (Browne, 2009) are two representative business rules management systems.

Business rules can be useful in many different ways. First, they can capture implicit and informal policies, thus providing an insight into business practices. Second, the analysis of current BRs can reveal gaps and problems in the way business is done, hence contributing towards business process improvement. Third, the analysis of BRs can support the development of information systems. Fourth, BRs can be integrated within workflow management systems for both design and execution purposes. BRs can trigger or constrain the execution of a business process

(Bajec and Krisper, 2005), and they can enable business process agility by supporting control flow decisions and process composition, as well as by enforcing and adapting constraints on BP data (Graml et al. 2008).

Business rules can be used in an SCM context to express SC policies and practices, and they can contribute in all four ways described above. We recognise the relation of BRs and SC workflow execution, and their potential for contributing towards SC agility, e.g. through guiding flexibility decision-making. There is limited research on the use of BRs for SCM, and most of relevant research papers focus on BR modelling. For example, Kim and Rogers (2005) propose an object-oriented five-view modelling framework, with the aim of improving flexibility for SC modelling. The framework includes the integration of business rules, and the authors describe a method for extracting business rules from the five-view model through event scripting.

2.4.2 Knowledge-Based Simulation

Knowledge-based techniques can drive and support simulation in several ways. Firstly, a simulation model can be represented in a declarative, knowledge-based fashion; this brings the benefit of specifying the structure of the model without worrying about how it should be run. Secondly, knowledge-driven simulation behaviours and results can be explained to the user, in the form of simple execution traces or deeper knowledge about the simulation model; this is particularly useful in the case of complex and dynamic systems, where simulation results are non-obvious. Thirdly, decision-making at simulation runtime can be enabled through a knowledge-based approach; this is valuable for dynamic domains, where adaptive and flexible behaviours are common. Further advantages of knowledge-based simulation can be found in Doukidis and Angelides (1994).

Research on knowledge-based simulation demonstrates the benefits discussed above in different domains. Robertson et al. (1991) propose a logic-based approach to ecological modelling, which exploits advantages within the first two categories. They represent ecological simulation models in Prolog, and they argue that a declarative approach brings benefits of explicit structure, modularity and flexibility of use. They recognise the importance of explanations in the domain of simulation

modelling, and they include an explanation facility in the implemented system, enabled through the declarative approach. Advantages of the third category are exploited by Aydemir et al. (2005) for a mechanical engineering problem, i.e. the efficient design of tube hydroforming processes. They suggest simulating a tube hydroforming process while updating parameters of the process at run time. The adaptive aspect of simulation is enabled through a fuzzy knowledge-based controller.

Knowledge-based simulation can potentially bring all the above benefits to an SCM setting: The specification of SC simulation models could be simplified for domain experts, explanations could be provided for simulation behaviours and results, while SC agility and flexibility aspects could be incorporated in the simulation model. However, to our knowledge there is no research investigating whether, and to what extent, these benefits hold in the case of the SCM domain.

2.4.3 Fault Diagnosis

Fault diagnosis concerns identifying the causal origins of abnormal or undesired events or situations. These causal origins are also called root causes or failures, while the abnormal events are also called faults or symptoms. Fault diagnosis is an important task, as it is a prerequisite for repairing and resolving problematic situations. The problem of diagnosis is relevant to several domains, such as medicine, finance, software engineering, mechanical engineering and process control. Venkatasubramanian et al. (2003) provide an extensive literature review on fault diagnosis in the process control domain and they present a generic classification scheme of diagnostic methods. This classification is based on the a priori knowledge used, which typically covers the set of failures and the relationship between symptoms and failures. Three main categories are identified: (1) quantitative model-based approaches, in which a priori knowledge has the form of mathematical functions, (2) qualitative model-based approaches, in which a priori knowledge has the form of qualitative functions and (3) process history-based approaches, in which there is no a priori knowledge available but such knowledge can be extracted from large amounts of relevant historical data.

Knowledge-based techniques are known to be useful for dealing with diagnostic tasks, and they are highly relevant to qualitative model-based approaches. Causal

relationships between failures and symptoms can be captured in a causal model, in the form of digraphs and fault trees (Venkatasubramanian et al. 2003). This causal knowledge can be expressed in the form of rules, thus guiding the diagnostic analysis (Isermann, 2006). Expert systems are also applicable to diagnostic problems, and it is worth noting that one of the most famous early rule-based expert systems, MYCIN (Shortliffe, 1976), was developed for the diagnosis and therapy recommendation for infectious diseases. The main advantage of a knowledge-based approach for diagnosis is its transparent reasoning and the ability to provide explanations on how the fault originated and propagated to the abnormal or undesired event. Deriving such explanations is heavily based on the reasoning process for identifying the root cause, and thus does not typically require much additional software development effort.

Diagnostic methods can be useful for analysing and repairing SC disruptions. In the last few years there has been increasing interest in SC disruption analysis, but research in this area is still in its infancy. Some research papers, such as Naim et al. (2002), propose managerial frameworks towards SC diagnosis, suggesting specific steps for guiding the discussion between SC managers. However, such an approach assumes participation, collaboration and trust between different SC members, and it is highly time-consuming. Hence, automated SC disruption analysis seems highly beneficial; research in this area is discussed in Chapter 3.

2.4.4 Relevance to the Project

Recognising the benefits of knowledge-based techniques discussed in this section, we adopt a knowledge-based approach for modelling and simulating SC operation. This way simulated behaviours and results can be explained. Furthermore, we employ business rules for formalising the thinking behaviour of SC members with respect to policies and flexibility decision-making. Finally, inspired by knowledge-based fault diagnosis, we specify a causal model of problematic SC operation, which drives SC disruption analysis.

2.5 Summary of Background

This chapter has introduced the main concepts needed as background knowledge for understanding the research presented in this thesis. An overview of Supply Chain Management was given, and desirable properties of a solution to the research problem were identified. We presented business process and workflow modelling techniques, emphasising their appropriateness for capturing SCM activities. A brief introduction to intelligent agents was provided, and the suitability of agent technologies for the SCM domain was also explained. Lastly, we discussed topics within the area of knowledge-based systems, such as business rules, knowledge-based simulation and fault diagnosis; knowledge-based techniques were found useful for analysing complex systems, such as supply chains. This chapter served as a gentle introduction to concepts and methods that are used in subsequent chapters. Work that is more specifically related to ours is discussed in the following chapter.

Chapter 3

Related Work

The problem of analysing supply chain operation dynamics has not been thoroughly addressed by existing literature so far. Nevertheless, we identify two research areas that are relevant to this problem: supply chain simulation and supply chain disruption analysis. SC simulation is relevant, as it provides an insight into SC-wide operation and allows the analysis of SC performance for different scenarios. SC disruption analysis is relevant to the research problem of this thesis, as it investigates the propagation of disruptive events across the supply chain. However, both research areas provide a partial solution to the problem, as specified in Section 1.1; SC simulation approaches cover mostly the first three issues of the problem, while SC disruption analysis approaches tackle the last two. This chapter presents a deep and narrow overview of related work in these two areas. Different approaches are explained, interesting aspects are discussed and research gaps are identified, thus highlighting the need for an intelligent solution to the research problem.

3.1 SC Simulation

Simulation is a useful technique for analysing supply chain operation, as "it can provide an insight into the operation of complex systems and can explore their behaviours" (Harrison et al. 2007, p.1243). There are several *advantages* of SC simulation. First, one can test and evaluate the effect of different decisions without actual implementation in the real supply chain. Second, simulation allows the study

of dynamic behaviours in the supply chain, as opposed to analytical methods. As Min and Zhou (2002, p.246) conclude from their literature review on SC modelling, "the resurgence of the simulation model is needed to evaluate dynamic decision rules for managing an inter-related series of supply chain processes". Third, one can gain an insight into the causes and effects of SC performance through what-if analysis. Fourth, simulation can deal with stochastic variables in the supply chain, and it is possible to check the impact of unexpected events on the entire supply chain. Last, simulation can also serve as a communicative means, encouraging stakeholders to exchange thoughts about alternative solutions to SCM problems. Given these benefits, simulation is regarded as one of the most powerful techniques for decision support in an SCM context (Terzi and Cavalieri, 2004).

Kleijnen (2005) identifies four *types* of SC simulation: (1) spreadsheet simulation, (2) system dynamics simulation, (3) discrete event dynamic system simulation and (4) business games. Even though spreadsheet simulation is not a formal and powerful simulation method, it is the most widely used method in practice (Chwif et al. 2002). System dynamics is a popular approach for studying complex systems. Its basic elements are feedback loops, stocks and flows, and the main idea is that levels of certain system variables are controlled by the rates of change of other variables. Angerhofer and Angelides (2000) provide a literature review on system dynamics simulation in SCM, and they highlight the successful application of this paradigm on the study of the bullwhip effect. Discrete event simulation captures the dynamic behaviour of a system, the state transition of which is guided by the occurrence of events. It is a powerful method that allows the extensive quantification of results and the incorporation of stochastic factors. Business games, such as the one proposed by Holweg and Bicheno (2002), are useful for educating and training users, as well as bringing SC managers together, thus supporting SC collaboration.

3.1.1 Commercial Approaches

There is a plethora of off-the-shelf SC simulators, as shown in Table 3.1. These are simulation tools tailored to the SCM domain with respect to modelling capabilities and performance measures. Most of these tools combine simulation and optimisation

techniques to help supply chain managers re-engineer their supply chains. They satisfy aspects of usability within the context of SCM practice and they typically provide powerful statistical analysis of simulation results. However, three main drawbacks are identified. Firstly, they do not provide an explanation of simulation results. This means that no additional information is provided to justify aspects of the simulation output, such as performed SCM activities and measured performance. Questions like the ones presented in Section 1.1.1 cannot be directly answered by existing simulation tools; this challenging task is instead left to the user. Secondly, the analysis of SC disruptions is not supported. In other words, the propagation of disruptive events along the supply chain cannot be tracked with the use of existing tools, and the causes of problematic situations cannot be automatically identified. Finally, SC agility aspects are usually neglected. This means that in many cases it is not possible to model and simulate highly flexible operations or decision-making; as a consequence, agile behaviours cannot be explicitly analysed.

IBM SC Analyzer (Bagchi et al. 1998; Archibald et al. 1999) is one of the most widely cited SC simulation tools. It combines optimisation and discrete event simulation techniques to analyse SC issues such as site location, manufacturing and transportation policies, as well as customer service. Optimisation is deployed to optimise the SC network's inventory before or during a simulation run. End-to-end SC simulation is enabled, thus a holistic new of SCM is adopted. SC Analyzer can model and simulate the following seven SC roles and functions: customer, manufacturing, distribution, transportation, inventory planning, forecasting and supply planning. The tool's outputs involve mainly cost, as well as fill rates, return rates, etc. SC Analyzer allows for graphical process modelling and animation, making it appealing to SCM practitioners. It has been applied to several supply chains of different industries, such as food and computer SCs. However, SC Analyzer does not provide an explanation facility, and the above-mentioned papers offer no account of SC disruption analysis. Moreover, agility aspects are not incorporated in the simulation model. Note that this tool is no longer available from IBM, as it has been sold to i2, which was later acquired by JDA.

	SC simulator	Input	Output	Simulation results explained	SC disruptions	SC agility
Commercial	SC Analyzer	customer, manufacturing, distribution, transportation, inventory planning, forecasting and supply planning	cost, fill rates, return rates, inventory, etc.	✘	✘	✘
	Supply Chain Guru	products, sites, demand, policies (sourcing, transportation, inventory policies)	financial reports, inventory units, customer service rates, resource utilisation rates	✘	✘	✘
	SmartSCOR	entities, products, resources, processes	static, dynamic, gap and causal analysis output	✘	✘	✘
	e-SCOR	SC roles, SCOR-based processes, process categories, atomic business process blocks	SCOR metrics	✘	✘	✘
	Supply Chain Builder	locations, items, inventory, shipments	unclear	✘	✘	✓
Research	Stefanovic et al. (2009)	supply network structure, process, business environment, constraints	SCOR metrics	✘	✘	✘
	Longo and Mirabellli (2008)	stores, plants, distribution centres, inventory policies	fill rates, on hand inventory, inventory costs	✘	✘	✘
	SCOR template	SCOR processes	SCOR metrics	✘	✘	✘
	Umeda and Zhang (2006)	types of SC members, activities, product flow strategies	order lead time, part inventory volume, part shortage rate, throughput	✘	✘	✘
	Easy-SC	enterprise nodes, functions, transportation paths, products, resources	unclear	✘	✘	✘

Table 3.1: Commercial and research approaches to SC simulation

Llamasoft Supply Chain Guru (LlamaSoft Incorporated, 2012) is another software tool that combines optimisation and simulation. The optimisation component can be used to determine the optimal structure and flow of products within an SC network, as well as the optimal inventory levels for the identified network and flows. The simulation component serves mainly as a validation of the proposed optimal SC design, and it can be used to predict and test the effects of the suggested SCM changes. The basic elements of a Supply Chain Guru model are the following: products, sites, demand, sourcing policies, transportation policies and inventory policies. Simulation output includes financial reports, inventory units, customer service rates and resource utilisation rates, which are visualised in sum-statistics and time series graphs. The visualisation capabilities of simulation input, run and output are regarded as considerable strengths of Supply Chain Guru, along with the statistical analysis of simulation results. Further advantages include its ability to incorporate variability and geographical aspects of SCM, the ease of importing data from spreadsheets and databases, as well as the support of what-if analysis through the specification of different simulation scenarios. Even though we consider Supply Chain Guru to be a powerful SC simulation environment, simulation results are not explained, and flexibility decisions are not incorporated in the model. Despite the rich output analysis, SC disruptions are not explicitly identified nor analysed.

IBM SmartSCOR (Dong et al. 2006; Ren et al. 2010) is a supply chain transformation platform that tackles the following two SCM problems identified in Chapter 2: SC configuration and SC operation. As far as SC configuration is concerned, optimisation techniques, such as mixed integer programming, are employed for SC network optimisation. As far as SC operation is concerned, SC process improvement is sought through process-oriented simulation and analysis. The basic elements of a simulation input model are the following: entities (i.e. customers, distribution centres, plants and suppliers), products, resources and processes. It is worth mentioning two points regarding the SmartSCOR simulation model. Firstly, the SCOR model is adopted for specifying processes across the supply chain. Secondly, all types of flows are considered between SC members (i.e. products, funds and information). Simulation in SmartSCOR is driven by IBM's WebSphere Business Modeller, a widely used software environment for business

process modelling and simulation. The use of WebSphere Business Modeller allows for rich static analysis (i.e. resource, organisation and general analysis) and dynamic analysis (i.e. aggregated, process instance and comparison analysis, as well as weighted average case analysis). In addition to these types of analysis, SmartSCOR is designed to facilitate two other types of analysis on SC performance simulation results: gap analysis and so-called causal analysis. Gap analysis involves comparing SC performance results to benchmarks with the use of spider charts. Causal analysis in SmartSCOR consists of what-if analysis, policy design through optimisation and root cause analysis. Root cause analysis in this case does not involve automated diagnosis, as discussed in Chapter 2, but instead the use of fishbone diagrams by business experts in order to assist them with the qualitative identification of root causes. Even though SmartSCOR recognises the need and usefulness of causal analysis for SC operation, the support it provides is limited. We regard this as the main shortcoming of this SC simulation tool. Furthermore, SC disruptions are not considered within SC operation analysis. As far as flexibility aspects are concerned, SmartSCOR allows scripting for special purpose simulation, but it is unclear whether and to what extent this includes SC agility.

Gensym e-SCOR (Barnett and Miller, 2000) is an SC modelling and simulation environment that adopts the SCOR model for SC planning purposes. There are four basic elements of an e-SCOR model, and they are hierarchically structured: SC roles (i.e. base manufacturer, manufacturer, distributor and consumer), SCOR-based processes (i.e. plan, source, make and deliver), process categories and atomic business process blocks. SC models are simulated in a discrete-event fashion, and simulation outputs are based on SCOR metrics. Similarly to other commercial SC simulation systems, e-SCOR is a visual tool that is easy to use; for example, simulation input can be specified with the use of drag-and-drop blocks. Furthermore, e-SCOR allows the definition of business rules, thus allowing the capturing of business logic of individual SC members at a lower level of detail. However, it does not support the explanation of simulation results, and SC disruptions are not taken into account.

SDI Supply Chain Builder (Phelps et al. 2000; Phelps et al. 2001; Siprelle et al. 2003) is an SC simulation platform that is part of a wider enterprise simulation

toolset. It employs discrete-event simulation techniques for analysing supply and distribution channels, and it can be integrated with SDI Plant Builder for studying entire supply chains. Its basic SC modelling blocks involve locations, items (i.e. materials and resources), inventory and shipments, thus covering consumption, ordering, order assignment, order filling and routing. We should note that dynamic aspects of SCM are considered, as alternative strategies can be defined for different SC conditions; this way, adaptive behaviours can be simulated. However, the papers do not discuss what type of simulation output is provided, how SC performance is measured and how simulation results are analysed.

3.1.2 Research Approaches

The largest body of research in SC simulation focuses on SC modelling issues, i.e. on capturing important aspects of SCM and on facilitating the specification of simulation input for SC managers. Most approaches propose generic SC modelling frameworks, and implement these with the use of general-purpose simulation platforms. Given that the focus is mostly on modelling aspects, the problems of explaining simulation results and analysing SC disruptions are not addressed, as shown in Table 3.1.

Stefanovic et al. (2009) develop an SC simulation environment by adopting a process-oriented approach that utilises the SCOR model. They identify four components of an SC model: supply network structure, process, business environment and constraints submodel. The authors claim that this modelling framework is generic (i.e. it can be applied to any type of supply chain) and close to reality (i.e. constraints on resources can be captured). The main component of the developed simulation software is a database that contains a process library and a collection of previously defined simulation models; this approach facilitates the process of specifying simulation input and allows the storage and querying of simulation results of different scenarios. However, the capabilities of this querying are not made clear in this paper, and the analysis of simulation results is not thoroughly discussed. Even though the paper mentions that business rules and policies can be defined for each SC member, it is not clarified whether these may

cover flexibility decision-making. Furthermore, SC disruptions seem to be outside the scope of this paper.

Longo and Mirabelli (2008) adopt a data-oriented approach for simulating supply chains, and they demonstrate it with the use of the discrete event simulation software eM-Plant. Their SC simulation model consists of three types of nodes (i.e. stores, plants and distribution centres) and four types of inventory policies. Instead of simulating this model with the use of library objects provided by eM-Plant, the authors implement appropriate methods in the scripting language provided by eM-Plant, and they propose the use of tables and event generators. They claim that this way a flexible, parametric and time-efficient SC simulator can be obtained. As far as simulation output is concerned, the following SC performance metrics are used for each SC node: fill rates, on hand inventory and inventory costs. For experimentation with different scenarios and what-if analysis, the authors propose the use of the simulator jointly with appropriate design of experiments and analysis of variance. Even though the analysis of variance is a useful statistical technique for testing the effect of certain simulation input parameters on SC performance, it does not provide an insight into how specific decisions and activities of certain SC members affect the behaviour and performance of other SC members and the SC as a whole. We should also note that SC agility and SC disruption aspects are not considered.

SCOR template (Persson and Araldi, 2009; Persson, 2011) is a set of SCOR-based building blocks in the general-purpose simulation software Arena. The objective of this research effort is to ease the process of specifying SC simulation input models for SCM practitioners. In order to achieve this, the authors utilise SCOR processes and metrics to define appropriate modules in Arena, which can be directly used by supply chain managers. The focus of this work is on the modelling procedure and usability, while simulation analysis is not discussed. To our knowledge, Arena does not provide an explanation facility; therefore, we believe that this is the case for simulation with the use of the SCOR template too. Furthermore, the authors do not discuss aspects of SC agility or disruptions.

Umeda and Zhang (2006) describe a generic SC simulation model which they implement in the general-purpose simulation environment Extend. They identify the following SC members: operational planner, parts supplier, products factory,

distribution centre, third-party logistics providers, consumers and retailers. They specify activities that each SC member performs and distinguish between three product flow strategies: push, pull and hybrid. The simulation output involves the following performance metrics: order lead time, part inventory volume at the product factory, part shortage rate at the product factory and throughput. Unfortunately, this paper does not discuss any advantages of the proposed modelling framework and there is no comparison to other approaches. Moreover, issues such as SC disruptions, SC agility and explanation of simulation results are not covered.

Easy-SC is an SC simulation tool for "studying the impact of stochastic demands, logistics decisions and production policies on key performance measures" (Liu et al. 2004, p.1374). In their paper, Liu et al. (2004) mainly describe the adopted modelling framework, consisting of enterprise nodes (i.e. suppliers, distribution centres, retailers, manufacturers, customers and carriers), the functions performed by enterprise nodes (i.e. source, make, deliver, inventory, transport and finance), transportation paths between enterprise nodes, products and resources. Simulation in Easy-SC is order-driven, and the tool includes policies and optimisation, which are not discussed in depth in the paper. Simulation output and analysis are not described in the paper, and SC agility and SC disruptions are not considered. The authors claim that Easy-SC allows for easy modelling through a graphical user interface and for easy extension, description and integration with other management information systems because it is implemented in Java. These claims, however, are not verified in the paper.

Another stream of research aims at facilitating the automatic generation of SC simulation models, so that users can correctly define the simulation input, even if they do not have advanced knowledge of modelling and simulation techniques. The work by Chatfield et al. (2006) and Cope et al. (2007) falls into this category. Chatfield et al. (2006) address issues of SC simulation specification, storage and model generation, and they present an appropriate object-oriented SC simulation tool called SISCO. SISCO users do not explicitly specify SC simulation models, but they provide SC descriptions, which are stored in an XML-based format called Supply Chain Modelling Language (SCML). The SCML-based descriptions are then translated to a Java-based object-oriented simulation model, which can be easily run.

Cope et al. (2007) and Cope (2008) adopt an ontology-based approach for automatically generating Arena simulation models that are suitable for stochastic, dynamic and distributed SC environments. They define a SCOR-based ontology, and implement an automatic model generator that parses the ontology and user-defined data on the simulation scenario to obtain a fully executable simulation model in XML. This XML file is then mapped to appropriate Arena modules, thus allowing the execution and analysis of the scenario. Since the objective of Chatfield et al. (2006) and Cope et al. (2007) is the automatic generation of SC simulation models, it is understandable that they do not focus on simulation analysis aspects. This means that they do not explain results and they do not deal with SC disruptions.

Other research efforts involve the simulation of specific types of supply chains. ALADIN (van der Vorst et al. 2009) is a simulation environment for food supply chains that is built on the discrete event simulation software platform Enterprise Dynamics. There are two main points of added value, as identified by the authors: Firstly, it allows the integrated modelling of food quality, sustainability and product logistics aspects. Secondly, control structures are explicitly modelled by considering decision-making agents, policies and interactions, as prescribed by the modelling framework of van der Zee and van der Vorst (2005). ALADIN demonstrates that generic SC simulation tools may in some cases not capture all important aspects of particular supply chains. However, advanced analysis of simulation results is not provided and the paper gives no account of SC agility aspects.

In Chapter 2 we presented agent-based approaches for simulating SC operation, such as Swaminathan et al. (1998), Allwood and Lee (2005) and Ivanov et al. (2010). Agent-based modelling is useful for capturing fundamental SC operations and decisions made by each SC member, while incorporating SC agility aspects. However, to our knowledge, the problem of analysing SC operation dynamics has not been addressed by this stream of research.

3.1.3 Gaps in SC Simulation

Existing SC simulation approaches have considerable strengths, especially with respect to usability. Commercial SC simulators provide graphical user interfaces and pre-defined SC building blocks, thus easing the process of specifying SC simulation

input. Animation is typically available during a simulation run, and simulation output can be statistically analysed and graphically visualised. Another aspect of increased usability involves the ability to integrate with other business information systems. Additionally, several commercial SC simulators incorporate variability and geographical aspects of supply chains. However, simulation is treated as a black box, and hence SC behaviours and simulation results are not explained. SC disruptions are not explicitly addressed by commercial SC simulators, and SC flexibility aspects are only partially considered by some.

Research on SC simulation consists of generic SC modelling frameworks that can be easily used by supply chain managers, while there are also some efforts towards the automatic generation of SC simulation models. However, the problem of analysing SC operation dynamics and SC disruptions seems to have been neglected by the research community. Furthermore, it is not clear to what extent SC agility is incorporated in such simulation efforts.

To summarise, we identify the following three gaps in related work on supply chain simulation:

- SC simulation results are not explained. This means that the problem of analysing SC operation dynamics is not directly addressed.
- SC disruptions are not analysed, and often they are not explicitly modelled. This means two things. Firstly, simulated SC behaviours and performance are not linked to the occurrence of disruptions. Secondly, the propagation of SC disruptions is not investigated, and the effect of SC disruptions on SC operation is not made clear.
- SC agility aspects are typically not incorporated in SC simulation models. This means that flexibility decisions and behaviours are not analysed as part of SC operation.

3.2 SC Disruption Analysis

There are two main approaches towards SC disruption management: the preventive and the interceptive approach. The preventive approach aims at reducing the likelihood of occurrence of disruptive events. To this end, robust strategies for mitigating SC risk are employed. This subject is typically studied by the Supply Chain Risk Management research community. The paper by Christopher and Lee (2004) falls into this category, while Tang (2006) provides an extensive literature review of the field. The interceptive approach aims at reducing the severity of effects of occurred disruptive events through SC monitoring and correction activities. This problem has given rise to the relatively new research area of Supply Chain Event Management. Illustrative approaches in this area include the ones by Bodendorf and Zimmermann (2005) and Bearzotti et al. (2012).

Understanding the causes and effects of disruptive events on SC operation and performance is crucial for both approaches presented above. Supply chain disruption analysis is, thus, a prerequisite for supply chain risk and event management. Research in this area is still in its infancy; a limited number of research papers have been published, all within the last five years. They propose the use of formal modelling techniques (i.e. variations of Petri Nets) for representing and analysing SC disruptions. Before discussing relevant work presented in Table 3.2, it is worth providing a brief introduction to Petri Nets.

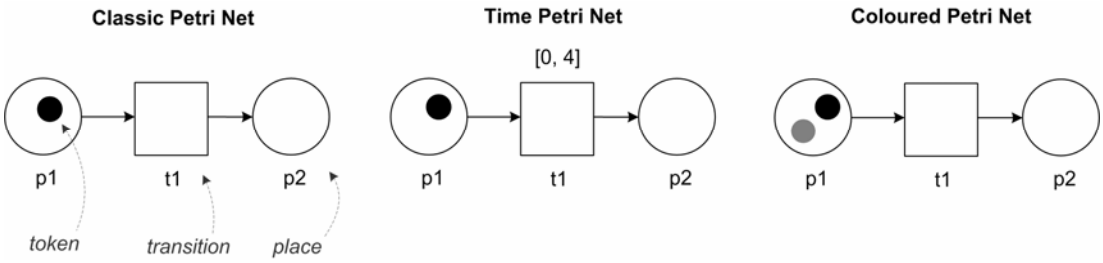

Figure 3.1: Illustrative classic Petri Net and variations of time and colour

A Petri Net (PN) is a graph that consists of places and transitions, which can be connected through directed arcs (van der Aalst and van Hee, 2004). Places may contain tokens, and the state of a Petri net is indicated by the distribution of tokens among its places, i.e. the token marking. Transitions can be fired, consuming tokens from the input place and producing tokens to the output place. This way tokens move

across the Petri net. A simple Petri net is shown in Figure 3.1. It is worth mentioning that Petri nets have formal semantics, allowing for powerful analysis. Apart from classic Petri nets, there are several variations, such as Time Petri Nets, Coloured Petri Nets and Time Coloured Petri Nets. Illustrative examples are presented in Figure 3.1. The time extension allows the description of temporal behaviours, such as durations of activities. The colour extension makes it possible to distinguish between different token types or values, and thus track behaviours for different case data.

		Liu et al. (2007)	Wu et al. (2007)	Zegordi and Davarzani (2012)
Causal analysis	Effect of disruption on disruption	✓	✓	✓
	Disruption root cause	✓	✗	✗
	Effect of disruption on performance	✗	✓	✓
	Performance root cause	✗	✗	✗
Different types of disruptions		✗	✗	✓
Tailored to SCM		✗	✗	✗
Maintainable		✗	✗	✗

Table 3.2: Characteristics of SC disruption analysis approaches

Liu et al. (2007) employ Time Coloured Petri Nets to study cause and effect relationships between disruptive events in a supply chain, as well as their effect on SC performance. They formalise normal and disruptive events as PN places, and event rules as PN transitions. They specify seven basic event patterns to capture event relationships (e.g. simple cause-result pattern, and N causes - 1 result pattern) that commonly arise in supply chains. A PN can be designed with the use of these event patterns as building blocks, thus modelling an SC scenario. Two types of analysis can be performed on SC scenarios that are modelled with this approach. Firstly, dependency graphs can be developed for a particular PN instance; these represent cause and effect relationships between events that were produced from the PN instance. This way the consequences and the root causes of disruptive events can be traced. Secondly, the developed PN can be simulated, allowing for the calculation of performance metrics, such as fill rates and average replenishment time of supply

orders. Through the use of simulation, what-if analysis can be performed to support SC improvement. The modelling approach proposed by Liu et al. (2007) is powerful, especially with respect to the analysis of causal relationships between disruptive events. The choice of Time Coloured Petri Nets allows the capturing of temporal relationships between events and the representation of a variety of case data. The main drawback of this approach is the fact that it is not sufficiently tailored to the SCM domain. The seven identified event patterns are of a technical nature, and there is no guidance for their use in order to model SC operation and disruptions. This means that SC scenarios are modelled on an ad hoc basis, thus limiting the usability of this approach for supply chain managers. Another weakness of this work involves maintainability with respect to the modelling procedure. Modifying aspects of an SC scenario requires considerable modification of the defined model. Given the example provided by Liu et al. (2007), modelling a scenario for a small supply chain can lead to a large PN representation; this fact makes the task of modifying the specified model even more complicated. The paper also does not discuss what type of SC disruptions can be modelled and whether causal relationships between different types of disruptions can be analysed. Finally, there is no explicit causal analysis between SC disruptive events and low SC performance.

Wu et al. (2007) propose a networked-based modelling approach, called DA_NET, to study the propagation of disruptions through a supply chain system, and calculate their effect on SC performance. DA_NET is a variation of Petri nets, in which attributes can be specified for place and transition nodes, and decision logic of transition nodes can be defined. DA_NET operations involve the firing of transitions based on the token marking and specified decision logic, thus leading to a new token marking in the network and an update of the corresponding attributes. Reachability analysis can be performed on DA_NET, identifying the set of place and transition nodes that can be reached from a certain initial token marking; for this reachability set it is possible to calculate the attribute update. In the context of SCM, DA_NET place and transition nodes can be used to model an SC scenario, while SC performance attributes (e.g. cost and lead time) can be specified for each node in the network. SC disruptions can be analysed in the following way: By placing a token at the place node that is disrupted, one can perform reachability analysis, and thus

identify how the SC disruption is propagated across the SC network. By calculating the attribute update for the identified reachability set, it is possible to measure the impact of the SC disruption on SC performance. One of the advantages of DA_NET is that it is possible to "analyse sub-networks, as long as the attributes of the place and transition nodes are set" (Wu et al. 2007, p.1669); this brings benefits of scalability. Unlike the approach proposed by Liu et al. (2007), this approach allows to explicitly identify the effect of an SC disruption on SC performance. However, it is not specified in the paper whether root causes of low SC performance can be identified, especially when there are several root causes of different types of SC disruptions. Similarly, DA_NET seems to support only the forward tracking of effects of SC disruption, while the backward tracing of root causes of SC disruptions is not covered. The main limitation of DA_NET is the lack of conceptual building blocks that are specialised for modelling supply chains; this makes the SC modelling process difficult for domain experts that do not have experience in PN modelling. Furthermore, there is no distinction between different SC disruption types, and interrelationships between different disruption types are not considered. In addition, the authors recognise the limitations of modularity and big size of the defined models; this fact raises maintainability issues.

Zegordi and Davarzani (2012) extend the approach of Wu et al. (2007) to deal with the last limitation. They employ Coloured Petri nets to distinguish between different SC disruption types and capture interrelationships between them. The proposed modelling approach, called CPND, closely follows the DA_NET approach, which is extended in two ways: Firstly, tokens of different colours exist in the network; different colours can be used for different SC disruption types. Secondly, the flow of coloured tokens through a transition node can change their colour or create new tokens of different colours; this way one can model relationships between different types of SC disruptions (e.g. one type of disruption gives rise to some other type of disruption). Reachability analysis can be performed similarly to Wu et al. (2007), thus tracking the propagation of SC disruptions and assessing their effect on SC performance. This work shares most of the strengths and weaknesses of DA_NET. The only difference is that CPND is sensitive to SC disruption types and

interrelationships between them. As in Wu et al. (2007), the main drawback is that it is not tailored to the SCM domain, thus perplexing the SC modelling process.

3.2.1 Gaps in SC Disruption Analysis

Related work in SC disruption analysis employs formal modelling techniques to represent SC operation scenarios and analyse the propagation of disruptions across the supply chain. The main strength of the approach proposed by Liu et al. (2007) is that causal paths between SC disruptions can be tracked, thus identifying root causes and effects of disruptive events. Wu et al. (2007) and Zegordi and Davarzani (2012) can identify the effects of SC disruptions on SC performance, which is a considerable advantage.

We identify the following three gaps in related work in supply chain disruption analysis:

- The proposed modelling methods are not sufficiently tailored to the SCM domain. This means that the process of SC modelling can be difficult for supply chain managers, thus limiting the usability of the suggested approaches.
- The identification of root causes of low SC performance is not covered. This is a considerable gap in the case where several SC disruptive events of different types lead to low SC performance.
- There are maintainability issues with respect to the supply chain modelling process. The lack of usable and modular SC modelling building blocks means that SC scenarios are modelled on an ad-hoc basis, leading to large and difficult to manage representations.

3.3 Conclusions

This chapter presented work in the areas of SC simulation and SC disruption analysis with respect to the research problem addressed in this thesis. Although there is an extensive body of work in SC simulation, the problem of analysing SC operation dynamics has been underdeveloped. Existing SC simulation approaches focus on

usability issues but do not explain simulation results. This is an important gap, as understanding the effect of SC decisions and activities of individual SC members on other SC members and overall SC performance is a prerequisite for SC improvement. SC agility aspects are typically not considered, and SC disruptions are not explicitly addressed.

Recent research efforts in SC disruption analysis provide a useful insight into the causes and effects of disruptive events. However, the adopted modelling approaches are not adjusted to the area of supply chain management and do not allow for modularity, thus raising issues of usability and maintainability. Furthermore, existing approaches do not allow the identification of root causes of low SC performance.

This thesis aims to fill these gaps and allow the analysis of SC operation dynamics for both normal and problematic SCM situations. To this end, a logic-based approach is adopted for modelling and simulating SC operation. Our declarative modelling framework is presented in the following chapter.

Chapter 4

Modelling Supply Chain Operation

Modelling supply chain operation for the purposes of this research should satisfy two requirements. First, the resulting models should cover the most important aspects of SC operation dynamics, as identified in Chapter 1, at both the local and the global level. Second, their formal representation should facilitate the explanation of the dynamics of the domain. The modelling approach presented in this chapter satisfies these requirements. We begin by discussing the boundaries of the studied problem area, clarifying what is not part of the domain (Section 4.1). We then conceptualise SC operation (Section 4.2) by identifying three categories of constructs: structural, behavioural and disruption-related. The resulting conceptual models are SCM-specific and non-technical, and they satisfy the first requirement. The second requirement is satisfied by adopting a knowledge-based approach for formalising SC operation (Section 4.3). Technical abstractions and declarative specifications of SC operation constructs are discussed. The conceptualisation and formalisation of an SC example is also presented, illustrating that the proposed approach is useful for describing SC operation and the dynamics involved.

4.1 Scope

As stated by the research hypothesis specified in Chapter 1, the systems that we study are supply chains, and we focus on their operational behaviour. We are interested in generic SC operational behaviour, and hence we do not limit this study

to some particular business sector (e.g. food SCs). There is also no limitation with respect to SC size and structure; instead we consider SCs consisting of any number of tiers of varying depth.

Basic entities in the SC system are the SC members, the market for the SC's final product and the products, funds and information moving across the supply chain. SC members are permanent entities in the SC system and they perform operational activities, which cause the flow of products, funds and information. Similarly, the market for the SC's final product is a permanent entity and it generates orders to the final nodes of the supply chain. On the contrary, products, information and funds are temporary entities in the SC, and they can be created, transformed or destroyed by SC members.

We consider the SC system as closed with respect to its environment. The SC environment is anything outside the studied supply chain and its flows, such as other supply chains (e.g. competing supply chains) and companies that are not directly related to the studied supply chain. The wider business environment, as shaped by political, economic, social, technological, environmental or legal factors is not taken into account, unless it directly affects SC operation.

We focus on operational aspects of supply chain management and the dynamics involved. Recognising that SC operation is affected by decisions on SC planning, demand forecasting and configuration, such decisions are implicitly included in this research. This means that the decisions on such matters are considered as input to SC operation, but the decision-making process on these issues and the related dynamics are beyond the scope of this research. For example, in order to study the operation dynamics of a particular SC, we take its configuration (i.e. the SC structure) into account, but we are not interested in the configuration procedure that led to this structure (i.e. the negotiation between different SC members during SC formation).

In order to study SC operation dynamics, one needs to take into account both global (i.e. SC-wide) and local (i.e. SC member-specific) aspects of the domain. Therefore, a varying level of detail is adopted for the study of SC operation, including e.g. high-level overall SC performance, lower-level production operations at some SC member and detailed information on an order placed by some SC member. For reasons of simplicity, we do not explicitly consider the organisational

structure and the number and locations of sites of SC members. We also regard softer business aspects, such as trust and culture, as beyond the scope of this research.

4.2 Conceptualising SC Operation

Having clarified the scope of SC operation and, thus, aspects that are not part of the domain's conceptualisation, this section presents the main constructs for conceptualising SC operation. These are classified into three categories: (1) structural constructs, which are things that exist in an SC and that are highly relevant to SC operation dynamics, (2) behavioural constructs, which describe the operational SCM behaviour of SC members and (3) disruption-related constructs, which are additional constructs, specialising on the description of problematic SC operation. We would like to make clear that this conceptualisation is not based on one research paper alone, but it has been developed given the wider theory of SCM, as presented in Chapter 2.

4.2.1 Structural Constructs

There are four main types of structural SC constructs: SC members, physical objects, information and events. *SC members* are the main actors of the SC, and they are understood as parties that add value to the SC. Examples of SC members are manufacturers, suppliers and wholesalers. Their behaviour, conceptualised in Section 4.2.2, drives SC operation.

There are different kinds of physical (material) objects that exist at some SC member or travel across the SC: products, resources and funds. The flow of *products* plays a crucial role in SC operation, as the supply chain's goal is the availability of the right products in the right place at the right time. There are several types of products at each SC member, such as raw materials, subcomponents, components and finished products. These are specified in the corresponding bill of materials (BOM), which is a list of parts along with the quantities of each needed to create an end product (Reid and Sanders, 2002). Products can be held as inventory at SC members and their lifecycle status can be: on order (i.e. items that have been ordered and are awaited for receipt), on hand (i.e. items that are available for use or sale),

reserved (i.e. items assigned to some received order) and in process (i.e. items that are assigned to some manufacturing activity and are under transformation to finished goods). Categorising inventory according to its purpose, we identify cycle inventory and safety stock; cycle inventory is the "average amount of inventory used to satisfy demand between receipt of supplier shipments" (Chopra and Meindl, 2003, p.57), while safety stock is inventory held to counter uncertainty (Chopra and Meindl, 2003). *Resources* are understood as equipment or machinery that is available at some SC member to support SC operation. As opposed to products, resources cannot be used up and they are not objects of exchange between SC members. They are typically characterised by some level of capacity, and their availability constrains SC operation. Examples of resources include transportation vehicles and production machinery. *Funds* flow across the SC (upstream) in return for the downstream flow of products. Their availability at some SC member is a prerequisite for the SC member's operational behaviour. There are three categories of funds at some SC member: receivable (i.e. funds that are awaited for receipt from some customer for a specific order), on hand (i.e. funds that are available for use) and payable (i.e. funds assigned for the payment of some placed order).

Information is available at each SC member and can be exchanged between SC members to support SC operation. It covers subjects such as orders (placed or received), sourcing or production lot sizes, SC partners (existing or prospective), etc. Certain information may be sensitive and, thus, only local (e.g. available funds), while other information may be happily exchanged in the form of messages (e.g. orders). Hence, the source of information at some SC member can be the SC member itself or other SC members. Distinguishing between the sources of information is useful, as SC members may decide to partially trust information received by other SC members. From now on we will call local information of the first type 'data', and transferred information of the second type 'facts'.

Events are incidents highly relevant to SC operation, and they can be the triggers but also the consequences of SC operation. They can occur at the global SC level (e.g. earthquakes) or at the local SC level (e.g. arrival of ordered items). Events occurring at some SC member may be internal (e.g. need for production) or external

(e.g. order receipt). Events typically give rise to SC members' operational behaviour, the conceptualisation of which is discussed in the following section.

4.2.2 Behavioural Constructs

Three facets of SC members' operational behaviour are identified: thinking, acting and interacting. *Thinking* refers to the decision-making process of SC members on operational matters. It may involve standard, routine decisions, such as when to place an order, or flexibility decisions, such as how to react to machinery breakage. This reasoning process can be simple or highly complex, and it can be based on predefined policies, best practices, recommended responses to specific situations or just common sense. As far as policies are concerned, we recognise sourcing and making policies as an integral part of SC members' thinking behaviour; these policies can be time- or quantity-based. It is worth mentioning two popular ordering policies: the (R,Q) policy, according to which a batch of size Q is ordered when the inventory position drops below R, and the (s,S) policy, which dictates that when the inventory position drops below s, an amount up to the maximum level S is ordered (Axsäter, 2006). SC members' thinking utilises existing information and drives their acting behaviour.

Acting refers to the extrinsic behaviour of SC members, which causes the flow of products, funds and information across the SC. As such, acting is the most important aspect of SC members' operational behaviour. We adopt the SCOR model (Supply Chain Council, 2008), as it is a widely accepted reference model of SC operation (Bolstorff and Rosenbaum, 2012). We, thus, recognise four areas of operational acting for each SC member: source, make, deliver and return. Note that SCOR's fifth area of 'plan' corresponds to the thinking behaviour of this conceptual model. We also support the varying level of detail proposed by the SCOR model, from the top to the configuration level, and from the process element to the implementation level. This way, acting behaviour can be captured at different granularity levels. For example, sourcing behaviour can be modelled at the top level as 'source', at a lower level as 'source stocked product' and at an even lower level as 'authorise supplier payment for sourced stocked product'. Note that we focus on the execution aspect of SCOR's configuration level, and we encompass all three suggested manufacturing

strategies: make to stock (i.e. make products based on demand forecasts), make to order (i.e. make products based on received customer orders) and engineer to order (i.e. design and make products based on customer specifications). It is worth clarifying that the SC members' acting conceptualisation utilises, but is not limited to the SCOR model; this means that we allow for richer or more specialised acting, if needed. Acting behaviour typically requires the availability of products, resources, funds and information, which are, as a result, transformed. Often an SC member's acting behaviour brings about his interacting behaviour.

Interacting refers to communication between SC members. As mentioned in the previous section, this involves the exchange of information in the form of messages. SC members may communicate as part of their standard order management behaviour or in order to deal with unexpected situations. Their interaction can be simple (e.g. inform about receipt of order) or highly complex (e.g. negotiate on changing ordered amount).

Apart from the three constructs of operational behaviour described above, we also identify behavioural meta-constructs on *SC performance*. As discussed in Chapter 2, performance measurement is an important aspect of SC operation, and managers wish to understand the dynamics of their SCs with respect to performance metrics. We use the SCOR-based framework for SC performance measurement (Supply Chain Council, 2008) for the reasons explained in Chapter 2, and recognise performance metrics along four SC performance attributes: reliability, responsiveness, cost and asset management. The fifth performance attribute proposed by the SCOR model, agility, is outside our conceptualisation scope, as it is more related to SC re-planning rather than operational activities, and it involves potential rather than actual behaviour (Beamon, 1999), thus its measurement is often assumption-based (i.e. contingency plans serve as the source of relevant data). We support the varying level of detail proposed by the SCOR model, from level 1 to level 3. This way, SC performance can be captured at different granularity levels. For example, cycle time can be modelled at the top level as 'order fulfilment cycle time', at a lower level as 'source cycle time' and at an even lower level as 'authorise supplier payment cycle time'.

4.2.3 Problematic SC Operation

We conceptualise problematic SC operation with respect to product flow across the SC, and we identify two main aspects: problematic situations that arise during SC operation and low SC performance. As far as the first is concerned, five types of problematic situations are identified: First, *delays* can occur at some SC member. These delays may involve any SCOR-based operational acting area, such as sourcing, making, delivering or returning. The delays may refer to the long duration of some acting behaviour, its late starting or its late completion. Taking these two dimensions into account, we can have source-start delays, make-finish delays, deliver-duration delays, etc. Second, *quality issues* can arise at some SC member, involving either resources or products that are available. Such examples are machine breakdowns, product damages and errors with items that lead to their destruction. Third, SC members can *act unusually*, possibly as a result of flexibility decisions that they make in the case of problematic situations. Such an example is the urgent sourcing from a non-standard supplier, i.e. the placement of a rush order to an alternative supplier which should be quickly fulfilled so as to avoid a stockout situation (Corbett, 2001). Fourth, *demand fluctuation* can take place, a typical example of which is the receipt of big orders (i.e. bigger than usually or expected). Fifth, *cancellation of order deliveries* can take place, which means that the delivery of some placed order can be cancelled by the corresponding supplier. Categorising these five types of problematic situations based on their source, the first three are experienced internally, the fourth is experienced through the demand side and the fifth through the supply side.

As far as *low SC performance* is concerned, this may involve any of the SCOR-based performance metrics, at any level of detail. SC performance is understood as low when the actual values of the metrics are beyond some threshold defined by the SC or the corresponding SC member. An example of low SC performance is cost that is higher than a certain value. Another example is perfect order fulfilment that is lower than some desired value. For reasons of simplicity, we focus on the following subset of cases of low SC performance: high cost, high cycle times, low on time rates.

These constructs can be used to specify problems that arise during SC operation. In order to facilitate the analysis of dependencies between such problems and to allow relevant explanation, we define *causal relationships* between them. We understand causal relationships in the following way: A causal relationship exists between two problematic situations if one is a possible reason for the other. For example, making can finish late because there is a "making duration delay". A "making duration delay" is a possible but not the only reason for the fact that making finishes late, as alternative reasons might exist. Moreover, the existence of a "making duration delay" doesn't necessarily lead to a "making finish delay", as some making tasks could be speeded up.

Before specifying such causal relationships it is worth explaining the general model, based on which these are derived. Any type of SC operational acting (e.g. making) can be considered as a process, as shown in Figure 4.1. This means that SC operational acting has a start and finish time, and it comprises of several steps. It also transforms specific inputs into outputs (both of which are usually materials/products) and it uses particular resources. Successful SC operational acting depends upon all these conditions, and hence it may finish late if any of these is not met. This means that SC operational acting may finish late if the required input or resources become available late, or if any of its constituent steps has a duration delay. The same resources can successively be used for SC operational acting, and thus a resource may become available late if it is released late by previous SC operational acting.

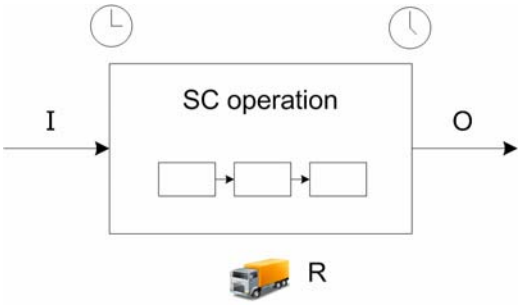

Figure 4.1: SC operational acting involves inputs I, outputs O and resources R, consists of several steps and has a start and finish time

According to the SCOR model, there are interfaces between different types of SC operational acting (e.g. between making and delivering), allowing the flow of products within and across SC members. Figure 4.2 depicts such interfaces in an

abstract form, highlighting that the output of some type of SC operational acting can serve as the input for the following type of SC operational acting (e.g. made products are used for delivering). This means that there is a dependency between the two, and hence the input required for some SC operational acting may become available late if the output of a preceding linked SC operational activity becomes available late. The same type of problem occurs if the amount of the involved output is lower than the input needed.

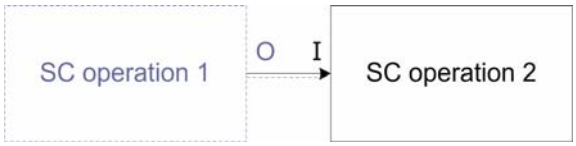

Figure 4.2: Different types of SC operational acting are linked, sharing their inputs and outputs

Based on the above general model, a set of causal relationships has been defined. We should clarify that our aim is not to provide an exhaustive list of causal relationships for problematic SC operation, but to capture the most important and typical ones. We focus on the SC flow of products as a result of sourcing, making and delivering, and on the dynamics involved. Resource constraints (e.g. machinery usage) are also taken into account. We assume that unusual behaviour is costly, which is a natural assumption to make. Given this scope, we identify the following causal relationships:

1. Sourcing finishes late because the needed sourced materials arrive late.
2. Sourcing finishes late because there is a sourcing duration delay.
3. The needed sourced materials arrive late because their delivery finishes late.
4. Making finishes late because the needed materials (i.e. production components) for making become available late.
5. Making finishes late because the needed resources for making become available late.
6. Making finishes late because there is a making duration delay.
7. The needed materials for making become available late because their sourcing finishes late.

8. The needed resources for making become available late because the previous making in which they were used finished late.
9. Delivering finishes late because the needed products for delivery become available late.
10. Delivering finishes late because the needed resources for delivery become available late.
11. Delivering finishes late because there is a delivering duration delay.
12. The needed products for delivering become available late because their making finishes late.
13. The needed products for delivering become available late because this delivery involves an unusually big order.
14. The needed products for delivering become available late because there is a shortage due to a previous unusually big order.
15. The needed resources for delivering become available late because the previous delivering in which they were used finished late.
16. Unusual acting takes place due to some other unusual acting.
17. Unusual acting takes place due to a flexibility decision.
18. A flexibility decision is made because of a problematic situation.
19. The special case of a problematic situation that involves the communication of a flexibility decision is due to that flexibility decision.
20. The cost is high because unusual acting takes place.
21. The on time rate is low because some orders are delivered late.
22. The cycle time is high because some duration-delays occur.

Most of these causal relationships are self-explanatory, and hence we will not further describe them. However, it is worth discussing the constructs of problematic SC operation that they refer to. Causal relationships 1-15 and 21-22 refer to delays, while 13 and 14 refer to demand fluctuations. The subject of causal relationships 20-22 is low SC performance, while 16, 17 and 20 refer to unusual acting. Causal

relationships 18 and 19 are more generic and potentially cover all constructs of problematic SC operation.

One can easily observe that there are interdependencies between the identified causal relationships. We will demonstrate this through two examples. The first example deals with causal relationships for delay-suffering SC operation. Suppose that making at some SC member finishes late. This can be because the needed materials for making become available late (given 4), which can be due to their late sourcing (given 7), and this fact can be due to their late arrival at the SC member (given 1). A possible reason for this late arrival is that their delivery by the supplier finishes late (given 3), which can be due to a delivering duration delay (given 11). Based on this chain of causal relationships, we can conclude that a possible reason for the making finish delay at some SC member is a delivering duration delay at his supplier. The second example deals with causal relationships for unusual SC operation. Suppose that a problematic situation (e.g. error with items) occurs at some SC member. This can lead to a flexibility decision (given 18), which can lead to unusual acting (given 17). Unusual acting is a possible reason for high cost (given 20). Based on this chain of causal relationships, we conclude that errors with items can result in high cost.

4.2.4 Conceptual Model Example

Let us now illustrate the presented conceptualisation approach through an example of a conceptual model of SC operation. We refer to the SC introduced in Chapter 1, which will be used throughout the thesis as a demonstrating case. We present the structural and behavioural constructs, and we discuss problematic operation for this supply chain.

The SC, presented in Figure 4.3, consists of eight main SC members across four tiers: Supplier1, Supplier2, Supplier3, Supplier4, Supplier5, Manufacturer, Retailer1 and Retailer2. There are three additional, secondary SC members that are in charge of shipping items for particular SC members: Transporter1, Transporter2 and Transporter3 (commissioned by Supplier1, Supplier2 and Supplier4, respectively). It is worth mentioning that Supplier5 is not a standard SC member; instead he acts as a backup supplier for Supplier4, accommodating urgent orders very quickly but costly.

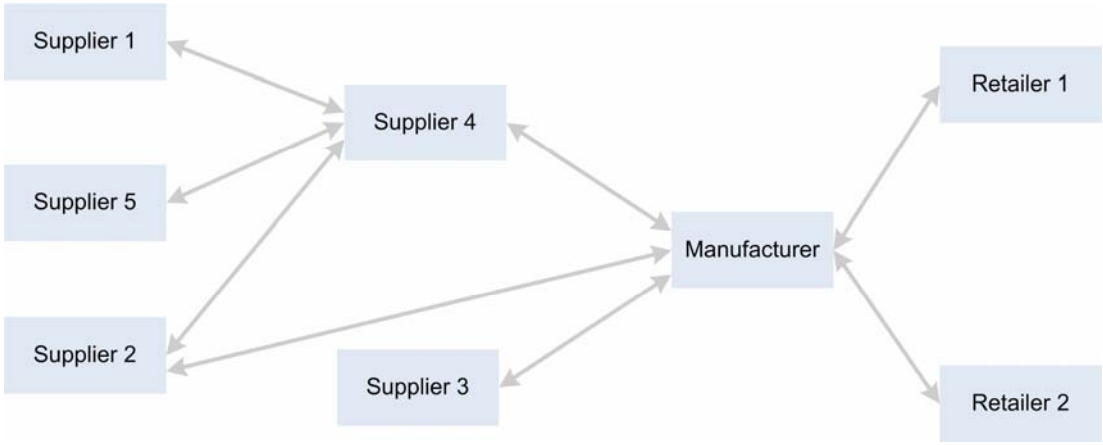

Figure 4.3: Example SC structure

There are five types of products that flow along the SC: Product1, Product2, Product3, Product4 and Product5. SC members provide one or more types of products to their customers. Table 4.1 summarises relevant information, along with information on the involved BOMs. The products that exist at some SC member as inventory have status of the types discussed in Section 4.2.1. For example, P1 items at Supplier4 can be on hand, in process or on order. Note that no safety stock is kept at any SC member.

SC member	Product	Bill Of Materials
Supplier1	Product1 (P1)	
Supplier2	Product2 (P2)	
Supplier3	Product3 (P3)	
Supplier4	Product4 (P4)	P4 = 1 × P1 + 4 × P2
Manufacturer	Product5 (P5)	P5 = 1 × P4 + 2 × P2 + 3 × P3
Supplier5	Product1 (P1), Product2 (P2)	

Table 4.1: Products provided by each SC member

As far as resources are concerned, Supplier1 has one machine for production, Supplier2 has three machines, Supplier3 has one truck, Supplier4 has one machine and Manufacturer has two machines and four trucks. Moreover, Transporter1 has one truck, Transporter2 has three trucks and Transporter3 has two trucks. As far as information is concerned, SC members keep information on the subjects discussed in Section 4.2.1. For example, Supplier4 keeps information on the following subjects: current suppliers for P1 and P2, urgent supplier for P1 and P2, transporter for

shipments, standard and urgent sourcing lot sizes for P1 and P2, manufacturing lot size for P4, placed orders, received orders, scheduled production activities and placed transportation requests. The events that occur at this SC are tightly coupled to SC operation. For example, the events that occur at Supplier4 are of the following types: need for standard sourcing of P1 or P2, need for urgent sourcing of P1 or P2, need for P4 production, scheduled order receipt, scheduled production and order receipt.

The SC members' thinking behaviour involves policies and flexibility decisions. The sourcing and making policies of SC members in this scenario are time-based. Supplier1 makes 6 P1 every 3 days, while Supplier2 makes 12 P2 every day. Supplier4 makes 6 P4 every 3 days, and sources 6 P1 every 3 days and 16 P2 every 2 days. Manufacturer makes 4 P5 every 2 days, and sources 4 P4 every 2 days, 8 P2 every 2 days and 18 P3 every 3 days. Retailer1 sources 3 P5 every 2 days, while Retailer2 sources 1 P5 every 2 days. It is worth mentioning that the flow rate of products across the SC is fairly stable and the timing and amounts involved in sourcing, making and delivering throughout the SC are well-tuned. Flexibility decision-making in this SC involves mainly reacting to errors with items. For example, Supplier4 decides to urgently source P2 items whenever there is an error with P2 on hand items. Moreover, Supplier4 decides to urgently source P1 items whenever he is informed about a cancellation of an order delivery for P1.

The SC members' acting behaviour involves sourcing, making and delivering. The SCOR-based processes that are relevant to this scenario are presented in Table 4.2, and they all involve stocked products (i.e. all SC members make to stock). Note that Table 4.2 also includes some special cases of processes: S1.1u and S1.24u involve urgent sourcing, and S1.24u is a combination of receiving and transferring sourced products. Figure 4.4 shows the corresponding processes for each SC member. The names of some processes in this figure include the involved product; for instance, Supplier4's S1.2p2 receives product P2. It is also worth mentioning that the process notation in this figure includes two numbers (at the right-hand corners of the process boxes). The number at the upper right corner refers to the process's cost, while the number at the lower right corner refers to its duration.

SC operation type	Process Code	Process Name
Source	S1.1	Schedule Product Deliveries
	S1.2	Receive Product
	S1.3	Verify Product
	S1.4	Transfer Product
	S1.1u	Schedule Urgent Product Deliveries
	S1.24u	Urgently Receive & Transfer Product
Make	M1.1	Schedule Production Activities
	M1.2	Issue Materials for Production
	M1.3	Produce
	M1.4	Package
	M1.6	Release Product
Deliver	D1.2	Receive Order
	D1.3	Reserve Inventory
	D1.11	Load Product on Vehicle
	D1.12	Ship Product
	D1	Deliver

Table 4.2: SCOR-based processes for example SC operation

The main interaction between SC members in this scenario is for order management reasons. This means that they send messages to place orders and make transportation requests, to inform about order deliveries, cancellations of order deliveries and fulfilled transportation requests.

We are interested in the following SC performance metrics for this SC: cost for each SC member (corresponding to SCOR's CO1.1 metric) and total SC cost, on time rate for each SC member (corresponding to SCOR's RL2.2 metric) and cycle time for each SC member's source/make/deliver operations (corresponding to SCOR's RS2.1, RS2.2 and RS2.3 metrics, respectively).

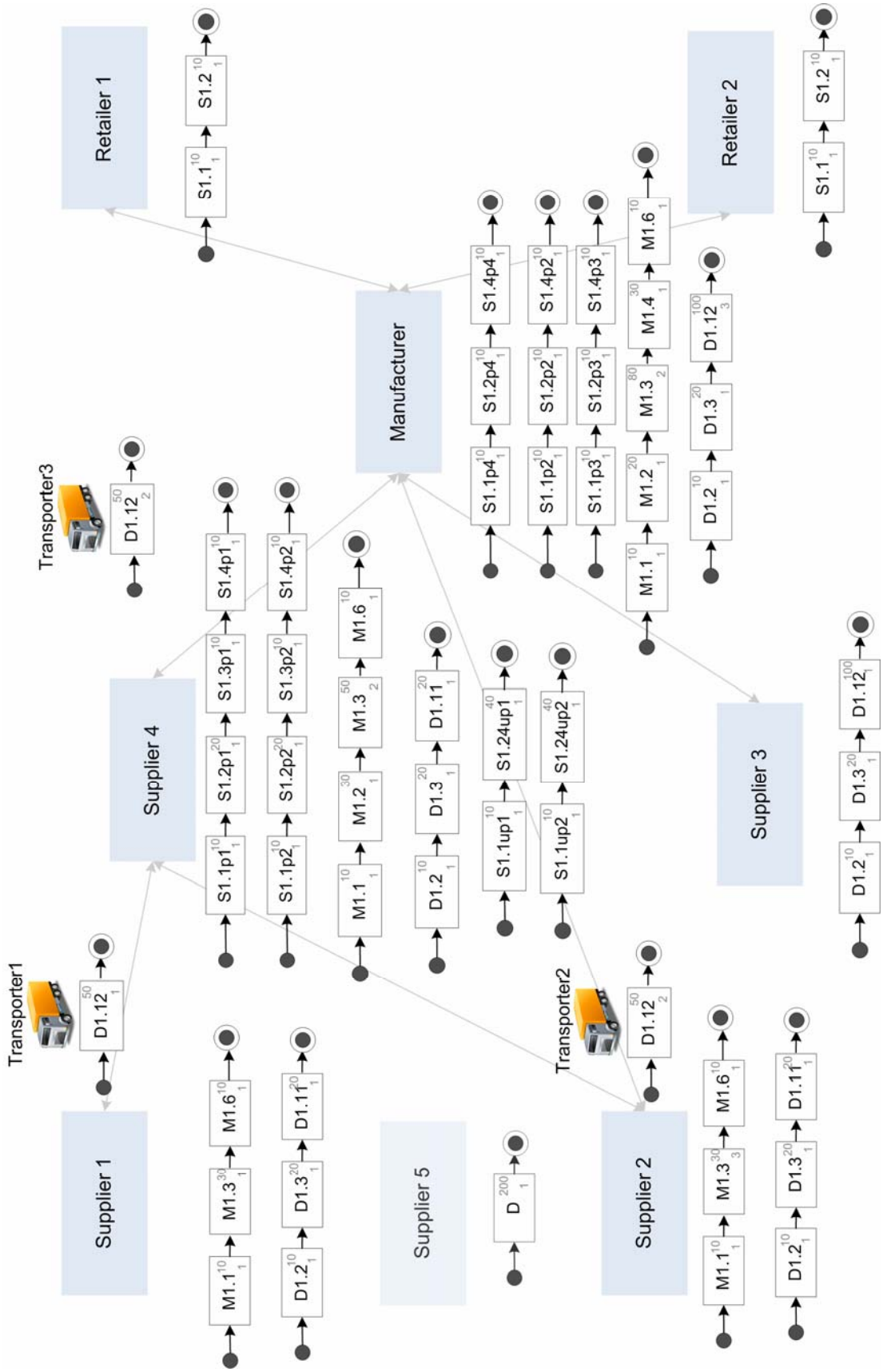

Figure 4.4: Business process models for example SC

As far as problematic SC operation is concerned, the following problematic situations can occur in this SC: delays, errors with items, big orders, cancellations of order deliveries and unusual processes. The types of unusual processes for this SC involve urgent sourcing and delivering, and they include Supplier4's S1.1u and S1.24u processes for P1 and P2 and Supplier5's D1 process. These problematic situations can cause low SC performance.

The SC operation dynamics of this scenario are complex, as there are several products flowing across the SC, and the SC structure implies several dependencies between SC members. For example, Supplier4 needs P2 items, which he sources from Supplier2. Hence, Supplier2's making policies and the resulting making behaviour affect the availability of P2 at Supplier4. And since Supplier4 uses P2 items for the manufacturing of P4, Supplier2's making behaviour indirectly affects the availability of P4 at Supplier4. This also means that a making delay at Supplier2 could be propagated to Supplier4, causing a making delay of P4, which could lead to delivery delays and, consequently, result into low on time rate for Supplier4. Another example of complex dynamics involves the indirect dependency between Supplier4 and Manufacturer with respect to P2. Since Supplier2 delivers P2 items to both Supplier4 and Manufacturer, the availability of P2 items at Supplier4 indirectly depends on Manufacturer's sourcing policies for P2.

The conceptual model of this scenario demonstrates that the conceptualisation constructs discussed in Section 4.2 are appropriate for describing SC operation in a sufficient way. The resulting conceptual model is simple to understand, and it focuses on particular aspects of the operation of this SC scenario (e.g. returns and the flow of funds are not considered). Nevertheless, it is comprehensive and powerful in two ways. First, it expresses the scenario's richness, including e.g. flexibility aspects and urgent sourcing. Second, it captures the scenario's operation dynamics, considering SC members' behavioural interdependencies with respect to product flow.

4.3 Formalising SC Operation

Having conceptualised SC operation, this section presents a knowledge-based approach for its formalisation. Structural, behavioural and disruption-related constructs are declaratively specified through Prolog-based predicates (Clocksin and Mellish, 2003), and illustrative examples are provided. The choice of a declarative formalism brings benefits of maintainability and reusability, a point that is further discussed in Chapter 6.

4.3.1 Structural Constructs

SC members are technically specified through *intelligent agents* (Wooldridge, 2001). There are three main reasons behind this decision. First, intelligent agents' characteristics of autonomy, social ability, reactivity and pro-activeness are highly relevant to SC members' behaviour during SC operation. This has been extensively discussed in Chapter 2. Second, an agent-oriented view of SC operation allows its study at two levels: the SC member-specific (which corresponds to an individual agent) and the global, holistic view of the SC (which corresponds to the multiagent system). This is particularly useful for analysing SC operation dynamics. Third, an agent-based abstraction of SC members is appropriate for capturing their operational behaviour, as conceptualised in Section 4.2.2. This means that an SC member's thinking, acting and interacting can be represented through corresponding intelligent agent layers. This is further explained in Section 4.3.2. The predicate-based definition of agent-oriented SC members is provided below, along with an example (in order to distinguish between the two, the definition is given in bold). Note that the following predicate is used to explicitly enumerate SC members; in order to further describe SC members (e.g. their policies or their state), separate definitions are needed. Such definitions are provided at the rest of Section 4.3 and they include a reference to agent-based SC members.

```
supply_chain_member(AgentId)
supply_chain_member(applessupplier)
```

Products and resources are *entities* that exist at some SC member at a certain timepoint, and they thus belong to the corresponding agent's local environment.

Their definition is entity-oriented and does not explicitly distinguish between products and resources. This generic approach is adopted, as it allows for economy when implementing the simulation environment, presented in Chapter 5. The predicate-based definition of such entities is provided below, along with one example for each (product and resource).

```
entity_occ(AgentId, EntityName, EntityId)
entity_occ(applessupplier, apple, prod_as23)
entity_occ(applessupplier, truck, res_as2)
```

In the case of products, it is important to specify *inventory* levels at some SC member at a certain timepoint. The predicate-based definition of inventory is provided below, and it captures information on the inventory status (as described in Section 4.2.1), its amount and the corresponding individual entities. The status can be on_hand, on_order, reserved or in_process. The form of ListOfEntityIds depends on the type of status, and there are three cases: (1) In the case of on_hand inventory definition, it includes only the list of their Ids, (2) in the case of on_order or reserved inventory definition, it includes the list of their Ids along with the involved OrderId, and (3) in the case of in_process inventory definition, it includes the list of their Ids along with the involved ProductionId. Two examples are provided to illustrate this point.

```
inventory(AgentId, Status, EntityName, EntityAmount,ListOfEntityIds)
inventory(applessupplier, on_hand, apple, 2, [prod_as23, prod_as24])
inventory(applessupplier, reserved, apple, 3,
  [prod_as13/order3, prod_as14/order3, prod_as15/order4a])
```

Other product-related concepts include *safety stock* and the *bill of materials*. Their predicate-based definitions are provided below, along with illustrative examples. Note that BOMList includes information on the components' entity names followed by the amount required for making one final product item.

```
safety_stock(AgentId, EntityName, SafetyStockAmount)
safety_stock(applessupplier, apple, 15)
bill_of_materials(AgentId, BOMId, EntityName, BOMList)
bill_of_materials(manufacturer, man_bom1, smoothie,
  [apple/5, banana/3])
```

The level of *funds* at some SC member at a certain timepoint is specified by distinguishing between the three funds' categories, discussed in Section 4.2.1: on hand, payable and receivable. The predicate-based definition of funds is provided below, along with two examples. Note that the ListOfOrderIds depends on the funds' category, and there are two cases: (1) In the case of on hand funds, it is empty, and (2) in the case of payable or receivable funds' definition, it includes the list of the amounts along with the involved OrderId.

```
funds(AgentId, FundsCategory, FundsAmount, ListOfOrderIds)
funds(applessupplier, on_hand, 12000, [ ])
funds(applessupplier, receivable, 3000, [2000/order3, 1000/order4a])
```

The *information* at some SC member at a certain timepoint is specified by taking its source into account, as discussed in Section 4.2.1. There are two broad categories of specification of information that is created locally by the SC member: generic and specialised. Locally created information is generically specified as data, and its predicate-based definition is provided below, along with an example.

```
data(AgentId, SubjectID, Content)
data(applessupplier, current_transporter, transporter5)
```

Specialised locally created information involves planned sourcing, making and delivering, and it includes placed and received orders, scheduled production and transportation requests. The predicate-based definition of such information is provided below, along with illustrating examples. These definitions are self-explanatory, and hence we will not further describe them.

```
placed_order(OrderId, AgentId, OrderingToAgentId,
  DestinationAgentId, EntityName, EntityAmount,
  ScheduledReceiptTime, ActualReceiptTime)
placed_order(order3, manufacturer, applessupplier, manufacturer,
  apple, 2, 10, 10)
received_order(OrderId, AgentId, OrderingAgentId,
  DestinationAgentId, EntityName, EntityAmount,
  ScheduledDeliveryTime, ActualDeliveryTime)
received_order(order3, applessupplier, manufacturer, manufacturer,
  apple, 2, 10, 10)
scheduled_production(ProductionId, AgentId, EntityName,
  EntityAmount, ScheduledProdTime, ActualProdTime)
```

```
scheduled_production(prod2a, applessupplier, apple, 5, 12, 12)
transportation_request(TransportRequestId, AgentId,
  TransportationAgentId, DestinationAgentId, EntityName,
  EntityAmount, ListOfEntityIds, ForOrderId)
transportation_request(transp108, applessupplier, transporter5,
  manufacturer, apple, 2, [prod_as13, prod_as14], order3)
```

Information received by other SC members is specified as facts, and its predicate-based definition is provided below, along with an example. Note that the facts' content may vary, covering subjects such as the successful delivery of an order or the cancellation of an order delivery.

```
fact(AgentId, Content)
fact(manufacturer, cancel_delivery(order3, apple, 2))
```

The predicate-based definition of *events* that occur at some SC member at a certain timepoint is provided below, along with an example. Note that InvokerId refers to the invoker of the event occurrence, such as a supply chain member or the ProductionId of a scheduled production, while the flag of an event links the event occurrence to a specific sourcing, making or delivering operation (more specifically, to the corresponding BPM instance, a point that is discussed in Section 4.3.2), if such a link exists.

```
event(AgentId, EventId, EventName, EventFlag, InvokerId, T)
event(applessupplier, e28, scheduled_production, bpm-e26, prod2a,12)
```

As far as the semantics of the formalised structural constructs are concerned, it is worth clarifying three points. First, the SC state at a certain timepoint can be thoroughly described through the constructs defined above (i.e. with respect to events, entities, information and funds available at different SC members). Second, constraints on the SC state can be a prerequisite for SC operation, and specifically for SC members' thinking, acting and interacting (e.g. a resource may be needed for acting). Third, the SC state is transformed through SC operation (e.g. interacting may update the information available at some SC member). The last two points are closely related to the execution semantics of behavioural constructs, and are thus further discussed in the following section.

4.3.2 Behavioural Constructs

In Section 4.2.2 we conceptualised that SC members think, act and interact. Mapping this conceptualisation to an agent-oriented representation (Wooldridge, 2001), we regard each SC member as an intelligent agent consisting of three layers:

- **Reasoning layer**: corresponds to the agent's ability to think and make decisions
- **Process layer**: corresponds to the agent's ability to execute processes, and thus act upon the environment
- **Communication layer**: corresponds to the agent's ability to receive and send messages, and thus interact with other agents

These three layers are tightly interconnected. Decisions made through the agent's reasoning layer are read by his process layer, thus triggering his acting behaviour. At the same time, processes executed through the agent's process layer change his environment, a situation which can initiate the agent's decision-making through his reasoning layer. Furthermore, the execution of processes through the agent's process layer can involve some communicative action, realised through his communication layer, while the receipt of some message through the agent's communication layer can be the prerequisite for the execution of certain processes through the agent's process layer. The agent's reasoning and communication layer are connected in a similar way: A decision made through the agent's reasoning layer might be communicated by utilising his communication layer, while the receipt of some message through the agent's communication layer might fire a decision-making process through his reasoning layer.

4.3.2.1 Business Rules

An agent's reasoning layer (and hence an SC member's thinking behaviour) is represented through *Business Rules* (The Business Rules Group, 2000). There are two main reasons behind this decision. First, BRs are a generic and expressive abstraction that can describe various types of principles that guide SC reasoning, such as policies, best practices and flexibility guidelines. Second, BRs are appropriate for guiding reasoning at different levels of detail and complexity; a

simple and concise BR can support simple decision-making, while the combination of several such BRs can effectively lead to highly complex decisions. A generic, declarative specification of a BR at some SC member is provided below.

br(AgentId, BrID, BrType, BrContent)

The form of a BR's content depends on its type, and we recognise three types of BRs for the context of SC operation: policies, flexibility BRs and process preconditions. The general form of a BR's content is the following: ifthen(IFpart, THENpart), where IFpart expresses the conditions of the BR, and THENpart its consequences. IFpart is a declarative expression, consisting of conjunctions and/or disjunctions of predicates, and it can be highly complex, if needed. THENpart is a list of consequences, which can be of reasoning, acting or interacting nature.

BRs for time- and quantity-based *policies* follow this formalism, and two examples are provided below. The first expresses a time-based production policy, according to which there is a need for production every 5 timepoints. The second expresses a quantity-based policy, according to which there is a need for production whenever on hand inventory drops below 20. The representation of BrContent within (R,Q) and (s,S) policies is more specialised and does not follow the general form described above. Instead the form of BrContent is rq_policy(EntityName, R, Q) and ss_policy(EntityName, SmallS, BigS) respectively.

```
br(applessupplier, br_as_m1, policy, ifthen(
  current_time_multiple_of(5), [create_event(need_for_production)]))
br(applessupplier, br_as_m2, policy, ifthen(
  less_or_equal(current_inventory(apple,on_hand),20),
  [create_event(need_for_production)]))
```

The technical specification of *flexibility BRs* follows the general form of BrContent. Conceptually, THENpart defines the reaction to the problematic situation described through IFpart. It is worth noting that flexibility-BRs are conceptually different from policy-BRs, but there is no computational difference between the two. This issue is clarified in Chapter 5, where we specify the execution semantics of BRs. We should also mention that we consider the explicit specification of flexibility business rules as a strength of our modelling framework, as this way SC agility aspects are incorporated. A flexibility BR example is provided below, which

specifies that urgent sourcing is needed if the current inventory reaches the safety stock level.

```
br(applessupplier, br_as_flex1, flexibility_br, ifthen(
  current_inventory_reaches_safety_stock_level(apple),
  [create_event(need_for_urgent_sourcing)]))
```

Business rules that serve as *process preconditions* follow the br/4 specification, but their BrContent does not follow the ifthen/2 form. Instead, it consists of one predicate, which can be negated if needed. Hence, this category of BRs differs from the previous two with respect to both conceptual and computational aspects. An example of precondition-BR is provided below.

```
br(applessupplier, br_as_pr3, precondition, \+ big_order(OrderId))
```

The execution semantics of the formalised business rules are discussed in Chapter 5. However, it is worth mentioning that a BR with content of ifthen/2 form is executed if the conditions expressed in its IFpart are satisfied, and its execution brings about the effects described in its THENpart. Let us now link these semantics to the semantics of the structural constructs that were formalised in the Section 4.3.1: The conditions of a BR involve the SC state (as described through the definition of structural constructs), and the effects of a BR's execution modify the SC state.

Business rules represent an SC agent's reasoning layer in a static way, while its decision-making process can be driven through a reasoning engine. Hence, a reasoning engine is used by agents to make decisions based on the defined BRs, and a logic-based implementation of such a reasoning engine is discussed in Chapter 6.

4.3.2.2 Business Processes

An agent's process layer (and hence an SC member's acting behaviour) is represented through *Business Processes*. There are three main reasons behind this decision. First, we conceptualised SC members' acting based on SCOR model's processes, which are naturally formalised through BPs. Second, BPs are suitable for capturing aspects of SC operational dynamics, given that their preconditions and postconditions are formally specified. Third, BP decomposition allows for description of SC members' acting behaviour at different levels of detail.

We recognise FBPML (Chen-Burger et al. 2002) as a useful foundation for formalising SC business processes, as it has formal semantics, it allows for the description of business process models with complex structure, and it facilitates their translation into executable workflows. The definitions presented in this section are an extension of previous work (Manataki, 2007) that followed FBPML. The declarative, predicate-based specification of a BP at some SC member is provided below. A process is executed if its preconditions (defined in PreconditionList) and trigger conditions (defined in TriggerList) hold. The execution of a process has the duration and cost defined in process/8 and it brings about the performance of the actions defined in ActionList. The execution semantics of the formalised business processes are further discussed in Chapter 5.

```
process(AgentId, Pid, PName, TriggerList, PreconditionList,
  ActionList, Duration, Cost)
```

An illustrating example of a BP definition is provided below. It is a process at applessupplier, and it involves producing apples. It is triggered for execution at the time of a scheduled production (and hence once a scheduled_production event occurs) and it is executed if there is information on this scheduled production and if one farmer is available. The execution of this process brings about the performance of the following actions: apples are created (of an amount as prescribed at the scheduled production) and they are added to the in_process inventory. The process is executed for 2 timepoints and costs 70 money units.

```
process(applessupplier, as_m13, produce_apples,
  [exist(event_occ(scheduled_production, ProductionId))],
  [exist(scheduled_production(ProductionId, apple, AppleAmount)),
   exist(entity_occ(farmer), 1, FarmerId)],
  [create_entity(internal, apple, AppleAmount, AppleIds),
   add_items_to_inventory(in_process/ProductionId, apple,
     AppleAmount, AppleIds)],
  2, 70).
```

We will now explain interesting arguments of process/8. A *trigger* is an event that occurs at the SC agent and that invokes process execution. There are two forms of trigger conditions: exist(event_occ(EventName, EventInvokerId)) and exist(event_occ(EventName)). The arguments of these predicates refer to event-related

information, as defined at event/6. If a process has no triggering conditions, then its TriggerList is of the form [true].

A *precondition* is a requirement for process execution which makes sure that its actions can be carried out successfully by the agent. Preconditions typically involve the availability of entities and information at some SC member. There are two forms of entity-related preconditions: exist(entity_occ(EntityName), EntityAmount, EntityStatus, EntityIds) and exist(entity_occ(EntityName), EntityAmount, EntityIds). The first form is mostly used for products (which have some inventory status), while the second is mostly used for resources. The produce_apples process example provided earlier involved an entity-related precondition of the second type. One precondition form is identified for the availability of funds: exist(funds(FundsAmount, FundsStatus)). There are five forms of information-related preconditions. Three of these involve placed orders, received orders and scheduled productions, while the other two involve the existence of data and facts at the SC agent. The respective predicates are provided below.

```
exist(placed_order(OrderId, OrderingToAgent, DeliveringToAgent,
  EntityName, EntityAmount, ScheduledReceiptTime))
exist(received_order(OrderId, OrderingAgentId, DestinationAgentId,
  EntityName, EntityAmount, ScheduledReceiptTime))
exist(scheduled_production(ProductionId, EntityName, EntityAmount))
exist(data(SubjectID, Content))
exist(fact(Content))
```

Apart from entity- and information-related preconditions, there are also BR-based preconditions. The specification of such preconditions within the process definition has the form br(BrID), where BrID is a business rule of type precondition that is defined through br/4.

A process's *action* is carried out when the process completes its execution, and it results into a modification of the world state. Actions typically transform, create or delete entities, funds and information, and they cause the occurrence of events. We identify four entity-related actions, the predicate-based representation of which is provided below. Actions create_entity/4 and create_entity_from_components/5 create a number of entities at the same or at a different SC agent, with the difference that create_entity_from_components/5 uses up the required components. Action

delete_entity/3 deletes a set of entities, while move_entity/4 involves the physical movement of a set of entities to some other SC member.

```
create_entity(ForAgentId, EntityName, EntityAmount, NewEntityIds)
create_entity_from_components(ForAgentId, EntityName, EntityAmount,
  ComponentIds, NewEntityIds)
delete_entity(EntityIds, EntityName, EntityAmount)
move_entity(EntityIds, EntityName, EntityAmount, DestinationAgentId)
```

Additional actions are identified for inventory management, thus transforming the inventory status of products kept in inventory. We identify six inventory-related actions, the predicate-based representation of which is provided below. The first four transform the inventory status of specific products, while the last two update inventory amount.

```
add_items_to_inventory(Status, EntityName, EntityAmount, EntityIds)
remove_items_from_inventory(Status, EntityName, EntityAmount,
  EntityIds)
update_items_status(FromStatus, ToStatus, EntityName, EntityAmount,
  EntityIds)
include_items_into_inventory(EntityName, Status, EntityIds)
increase_inventory_amount(EntityName, Status, EntityAmount)
decrease_inventory_amount(EntityName, Status, EntityAmount)
```

Four funds-related actions are identified, and their predicate-based representation is provided below. The first two involve the flow of funds for a satisfied order, while the rest involve their local status update.

```
make_payment(FundsAmount, OrderId)
receive_payment(FundsAmount, OrderId)
reserve_payable_funds(FundsAmount, OrderId)
add_receivable_funds(FundsAmount, OrderId)
```

We identify six information-related actions, the predicate-based representation of which is provided below. The first two involve the transformation of data, while the rest involve placed and received orders, scheduled productions and transportation requests.

```
create_data(SubjectID, Content)
delete_data(SubjectID)
```

```
place_order(OrderId, OrderingToAgent, ToDeliverAtAgent, EntityName,
  EntityAmount, ScheduledReceiptTime)
record_received_order(OrderId, OrderingAgentId, DestinationAgentId,
  EntityName, EntityAmount, ScheduledReceiptTime)
schedule_production(ProductionId, EntityName, EntityAmount)
create_transportation_request(TransportRequestId,
  TransportationAgentId, DestinationAgentId, EntityName,
  EntityAmount, EntityIds, ForOrderId)
```

The last actions' category involves the occurrence of events. These events can occur at the SC agent carrying out the action or at some other SC agent, and they can take place at the end of process execution or at some later scheduled time. Four such actions are identified, the predicate-based representation of which is provided below. Note that the occurrence of assigned events means that the events are allocated to a specific SC operation BPM through their flag.

```
create_event(AtAgentId, EventName, EventInvokerId)
schedule_event(AgentId, EventName, EventInvokerId, ScheduleT)
create_assigned_event(internal, EventName, EventInvokerId)
schedule_assigned_event(internal,EventName,EventInvokerId,ScheduleT)
```

One can see that both the conditions and the effects of business process execution involve the SC state, as described through the definition of structural constructs. Trigger conditions involve the occurrence of events, while preconditions involve the availability of entities, funds and information. Similarly, the actions that are performed as effects of a BP's execution involve events, entities, funds and information, and thus modify the SC state.

So far in this section we have discussed the declarative representation of an SC agent's processes. However, a complete and precise, formal model of an SC agent's process layer should also include the junctions in the involved business process model (BPM). As discussed in Chapter 2, junctions describe the control sequence of the BPs in the BPM, and FBPML suggests a wide range of such connectives. For the purpose of this research, the adopted junction types are: start, finish, link, and-joint, or-joint, and-split and or-split. The predicate-based specification of a junction within an SC member's BPM is provided below. JType refers to the junction type, PreList is the list of processes or junctions that are directly preceding the junction, while PostList is the list of processes or junctions that are directly following it. A junction is

executed if its execution semantics are satisfied, according to FBPML specification; this issue is further discussed in Chapter 5. A junction visual example and its declarative specification are following.

junction(AgentId, Jid, JType, PreList, PostList)

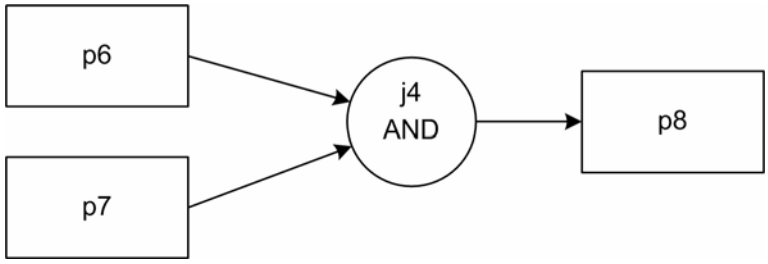

Figure 4.5: Visual representation of a junction

```
junction(applessupplier, j4, and_joint, [p6, p7], [p8])
```

As already mentioned, the definitions presented in this section are an extension of previous work (Manataki, 2007). Let us now clarify the context of the previous work, as well as the size and the content of the extension provided for this PhD project. In (Manataki, 2007) the development of a workflow engine for executing simple business process models within a single agent was presented. The domain representation in this earlier work included five constructs: entities, data, events, junctions and processes. For the purpose of this PhD project, the domain representation has been extended along four main lines. First, the specification of these five constructs has now been enriched to address aspects of the multiagent setting. Second, in (Manataki, 2007) we assumed that junctions are preceded or followed only by processes. We have now relaxed this assumption to allow junctions to be also preceded or followed by other junctions. This way more advanced BPM structures can be modelled. Third, the specification of business process preconditions and actions with respect to entities and events has been enriched to accommodate entity amounts and create or schedule events. Fourth, process preconditions and actions now address aspects of a richer domain representation, including placed and received orders, scheduled productions, transportation requests, facts, messages, inventory and business rules.

The formalisation approach discussed in this section is useful for representing SC members' acting behaviour; SCOR-based processes of different levels can be

formally represented through appropriate business process predicates, and their control sequence within a BPM can be specified through the declarative specification of junctions. A declarative approach is adopted in order to facilitate the capturing and explanation of the dynamics involved. It is worth mentioning that BPMs represent an SC agent's process layer in a static way, while its real-time acting behaviour can be driven through a workflow engine. Hence, a workflow engine is used by agents to execute processes based on the defined BPMs, and a rule-based implementation of such a workflow engine is discussed in Chapter 5.

4.3.2.3 Communicative Actions

An agent's communication layer (and hence an SC member's interacting behaviour) is represented through *communicative actions*. These actions involve sending and receiving messages. The declarative specification of messages is provided below, along with an example. Sender refers to the agent that sends the message and ReceiversList refers to the agents to which the message is addressed. A message can be a reply to a previous message (as denoted at InReplyTo), and it can be characterised by a Performative such as inform, refuse, propose, etc.

```
message(MessageID, Sender, ReceiversList, InReplyTo, Performative,
  Content, T)
message(mes23, applessupplier, [manufacturer], none, inform,
  cancel_delivery(order3, apple, 2), 12)
```

Two basic communicative actions are identified, send_message/4 and receive_message/1, and their formal representation is provided below. Note that certain BPs of an SC agent can involve the sending of messages, and therefore send_message/4 is considered as an action type additional to the ones presented in the previous section. The execution semantics of these two communicative actions are discussed in Chapter 5.

```
send_message(ReceiversList, InReplyTo, Performative, Content)
receive_message(MessageID)
```

4.3.2.4 SC Performance

Supply chain performance is measured during SC operation, and its formalisation follows the general form of performance_metric(AgentId, Value). The formalisation of selected SCOR-based metrics is provided in this section, while the formulas for their calculation are discussed in Chapter 5. Note that we declaratively specify a subset of the SC performance metrics that were conceptualised in Section 4.3.2.4; this is the subset of performance metrics that have been implemented within our simulation system: on time rate, cycle time, cost and overall SC cost.

```
on_time_rate(AgentId, Value)
cycle_time(AgentId, source, Product, Value)
cycle_time(AgentId, make, Product, Value)
cycle_time(AgentId, deliver, Product, Value)
total_sc_cost(Value)
cost(AgentId, Value)
```

4.3.3 Problematic SC Operation

In Section 4.2.3 we conceptualised problematic SC operation through problematic situations and low SC performance, and we defined causal relationships between them. The formalisation of these aspects is discussed in this section.

4.3.3.1 Constructs

Problematic SC operation is formalised in a declarative way. The formalisation of SC operational problems is presented in this section, while the reasoning for detecting such problems is discussed in Chapter 5. The predicate-based specification of the following problematic situations is provided below: process delays (duration-, start- and finish-delays), errors with items, unusual processes, big orders and cancellations of order deliveries. Note that ProcessInst refers to a particular instance of an SC agent's process that is executed.

```
process_duration_delay(ProcessInst)
process_start_delay(ProcessInst)
process_finish_delay(ProcessInst)
error_with_items(AgentId, EntityName, EntityAmount, EntityStatus, T)
unusual_process(ProcessInst)
```

```
high_demand(OrderId, ProcessInst)
delivery_cancelation(OrderId)
```

The formalisation of low SC performance, as discussed in Section 4.2.3, is provided below:

```
high_total_sc_cost(TotalCost)
high_cycle_time(AgentId, SCORtype, Product, CycleTime)
low_on_time_rate(AgentId, OnTimeRate)
```

4.3.3.2 Causal Model

The causal relationships presented in Section 4.2.3 are now formalised into a logic-based causal model. The general form of specifying a causal relationship between two problematic situations A and B is possible_reason(A, B), which means that B is a possible reason for A. The declarative specification of the identified causal relationships follows this general form and it is provided in Table 4.3. Note that the numbering corresponds to the numbering of causal relationships in Section 4.2.3.

It is worth clarifying three points on the defined causal model. First, many of the predicates used to refer to problematic situations in the causal model (i.e. A and B within possible_reason(A, B)) are different from the predicates defined in the previous section. Nevertheless, they are mapped or directly related to the predicates of problematic SC operation presented in the previous section. For example, the causal model includes the problematic situation make_finish_delay(MProcInst), which is a special case of the formalised construct process_finish_delay(ProcessInst) of Section 4.3.3.1. Another example is needed_source_material_not_available(S12ProcInst), which is mapped to process_start_delay(ProcessInst) of a corresponding SCOR-based process instance, such as S1.2 for a make-to-stock product or S2.2 for a make-to-order product.

Second, when using the causal model to explain problematic SC operation, one should check whether the individual problematic situations mentioned in the causal model hold. The point discussed above is useful for this task. For example, when using the first causal relationship to explain late sourcing, it should be checked whether needed_source_material_not_available(S12ProcInst) holds. For a make-to-stock product, this means checking whether there is a process-start-delay of a S1.2-process instance, which means checking whether process_start_delay(S12ProcInst) holds. The

definition of such rules is not part of the causal model and it is discussed in Chapter 6.

Causal Relationship	Formal Representation
1	possible_reason(source_finish_delay(SProcInst), needed_source_material_not_available(S12ProcInst)).
2	possible_reason(source_finish_delay(SProcInst), some_bpm_process_duration_delay(ProcInst)).
3	possible_reason(needed_source_material_not_available(S12ProcInst), deliver_finish_delay(DProcInst)).
4	possible_reason(make_finish_delay(MProcInst), needed_make_material_not_available(M12ProcInst)).
5	possible_reason(make_finish_delay(MProcInst), needed_make_resources_not_available(M13ProcInst)).
6	possible_reason(make_finish_delay(MProcInst), some_bpm_process_duration_delay(ProcInst)).
7	possible_reason(needed_make_material_not_available(M12ProcInst), source_finish_delay(SProcInst)).
8	possible_reason(needed_make_resources_not_available(M13ProcInst), make_finish_delay(PreviousMProcInst)).
9	possible_reason(deliver_finish_delay(DProcInst), needed_deliver_material_not_available(D13ProcInst)).
10	possible_reason(deliver_finish_delay(DProcInst), needed_deliver_resources_not_available(D111ProcInst)).
11	possible_reason(deliver_finish_delay(DProcInst), some_bpm_process_duration_delay(ProcInst)).
12	possible_reason(needed_deliver_material_not_available(D13ProcInst), make_finish_delay(MProcInst)).
13	possible_reason(needed_deliver_material_not_available(D13ProcInst), unusually_big_order(DProcInst)).
14	possible_reason(needed_deliver_material_not_available(D13ProcInst), unusually_big_order(PreviousDProcInst)).
15	possible_reason(needed_deliver_resources_not_available(D111ProcInst), deliver_finish_delay(PreviousDProcInst)).
16	possible_reason(unusual_process(UnusualProcInst), unusual_process(OtherUnusualProcInst)).
17	possible_reason(unusual_process(UnusualProcInst), flexBR_fire(FlexBR,T)).
18	possible_reason(flexBR_fire(FlexBR, T), problematic_situation(Situation, AgentId, T)).
19	possible_reason(problematic_situation(just_got_informed(Problem1), Agent1, T1), flexBR_fire(FlexBR, T)).
20	possible_reason(high_sc_cost, unusual_processes(ListOfUnusualProcInst)).
21	possible_reason(low_on_time_rate(AgentId), late_delivered_orders(AgentId, ListOfLateDeliveredOrders)).
22	possible_reason(high_cycle_time(AgentId, SCORtype, Product), duration_delayed_processes(ListOfDelayedProcInst)).

Table 4.3: Causal model of problematic SC operation

Third, when using the causal model to explain problematic SC operation, and given that the task mentioned in the previous paragraph has been done, one should also

check whether the specified problematic situations are actually related. For example, when using the first causal relationship to explain late sourcing and if a specific S1.2 process instance has been identified such that needed_source_material_not_available(S12ProcInst) holds, it should be checked whether this particular process instance is related to the sourcing finish delay. The definition of such rules is again not part of the causal model and it is discussed in Chapter 5.

4.3.4 Formal Model Example

We will now illustrate the formalisation approach discussed in this chapter through an appropriate example. We refer to the SC conceptualised in Section 4.2.4 and we present its formal model though the definition of structural and behavioural constructs. This formal model will be used in Chapter 5 in order to simulate the operation of this SC, and for this reason the formal model of structural constructs refers to the SC's initial state (i.e. at timepoint 0). Behavioural constructs, on the other hand, are not modified during simulation, and hence their formalisation is valid for any SC state. Illustrative examples of formalised problematic operation for this SC will also be provided. Note that due to limited space a subpart of the SC's formal model is provided here; this includes constructs that are specific to Supplier4. SC member Supplier4 is specified below.

```
supply_chain_member(supplier4).
```

The formalisation of structural constructs for Supplier4 at timepoint 0 is the following. Note that since we are referring to timepoint 0, some constructs might not be relevant, such as information on received orders and events. Some entities and inventory available at Supplier4 are declaratively specified below, and the bill of materials for Product4 follows. As far as the execution semantics of these constructs are concerned, it is worth clarifying that the two on hand Product4 items will be used during simulation to satisfy some incoming customer order (i.e. will be used to satisfy the preconditions of the corresponding D1.3 business process).

```
entity_occ(supplier4, machine, r_sup4_1).
entity_occ(supplier4, product4, r_sup4_2).
entity_occ(supplier4, product4, r_sup4_3).
inventory(supplier4, on_hand, product4, 2, [r_sup4_2,r_sup4_3]).
```

```
bill_of_materials(supplier4, sup4_bom1, product4, [product1/1,
  product2/4]).
```

A subset of the information available at Supplier4 at timepoint 0 is declaratively specified below, covering aspects like SC partners and lot sizes. The information on the production lot size for Product4 will be used during simulation for production scheduling; this point is further discussed at the specification of Supplier4's business process sup4_m11.

```
data(supplier4, product4_production_lot_size, 6).
data(supplier4, currenttransporter, transporter3).
```

The formalisation of behavioural constructs for Supplier4 follows. Illustrative business rules for policies and flexibility at Supplier4 are firstly provided. The defined BR br_sup4_3 specifies a time-based policy for Supplier4's making of P4. The execution of this BR causes the occurrence of an event of type need_for_production, leading to production scheduling; this point is further discussed at the specification of Supplier4's business process sup4_m11. Note that a time-based condition of the form A×K+B is satisfied every A timepoints after timepoint B, and specifying a timepoint B is useful for simulation purposes. The specified flexibility BR br_sup4_urg1 involves Supplier4's reaction to errors with P2 on hand items.

```
br(supplier4, br_sup4_3, policy, ifthen(
 current_time_form_of(3*k+15),[create_event(need_for_production)])).

br(supplier4, br_sup4_urg1, flexibility_br, ifthen(
 error_with_items(product2, EntityAmount, on_hand),
 [create_event(need_for_product2_urgent_sourcing),
  update_lot_size_if_needed(product2_urglot_size,EntityAmount)])).
```

A part of Supplier4's BPM for making Product4 is declaratively specified below. Figure 4.6 presents the corresponding BPM, as introduced in Figure 4.4, and the formalised junction and business process are marked in red. Let us explain what the predicate for process sup4_m11 means: This process involves scheduling the production for Product4. It is triggered by an event of type need_for_production, which occurs due to the execution of making policy BR br_sup4_3. Information on the production lot size for Product4 is needed for this process to execute (remember that the specification of this data was formalised earlier). The execution of this process

schedules the production of Product4 in the amount dictated by the lot size. There are two effects of the process's execution: information on the scheduled production is created and an event of type scheduled_production occurs. It is worth noting that this event will trigger the execution of the process following sup4_m11, i.e. sup4_m12.

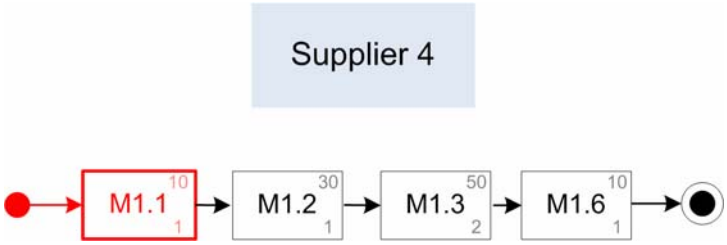

Figure 4.6: Supplier4's making BPM

```
junction(supplier4, sup4_jm0, start, [], [sup4_m11]).

process(supplier4, sup4_m11, schedule_product4_production,
  [exist(event_occ(need_for_production))],
  [exist(data(product4_production_lot_size, Product4Amount))],
  [schedule_production(ProductionId, product4, Product4Amount),
   create_assigned_event(internal, scheduled_production,
    ProductionId)],
  1, 10).
```

Supplier4's communicative actions involve sending and receiving messages for sourcing and delivering. The definition of process sup4_d111 involves the sending of a message for making a transportation request.

During SC operation (and thus during simulation) SC performance is measured and problematic situations are tracked. Illustrative examples of relevant formalisation are provided below.

```
cycle_time(supplier4, source, product1, 4).
process_start_delay(bpm-745/sup4_m12).
```

4.4 Modelling Summary

This chapter presented a formal, declarative approach for modelling SC operation. We conceptualised the domain by taking into account structural and behavioural constructs, as well as problematic SC operation. The conceptual model addresses the main aspects of SC operation dynamics, as identified in Section 1.1, in the following

three ways: First, the model covers all three dimensions (i.e. decision, actions and interactions) of SC members' operational behaviour that were recognised in Section 1.1. Second, SC performance is modelled at both the local and the global level. Third, the SC operation model includes constructs for SC disruptions; this is a considerable advantage compared to existing simulation models, as discussed in Chapter 3. It also incorporates flexibility aspects, which are increasingly important in modern supply chain management, and which are typically not incorporated in SC simulation models. Finally, we should emphasise that given this conceptualisation, the resulting model is tailored to the SCM domain.

In this chapter we also formalised the domain by regarding SC members as logic-based intelligent agents consisting of three layers: (1) reasoning layer, represented through business rules, (2) process layer, represented through business processes and (3) communication layer, represented through communicative actions. Structural and disruption-related constructs were declaratively formalised, and a logic-based causal model was defined, capturing possible reasons for the occurrence of problematic situations. We believe that the formal model achieves a good balance between conciseness and expressivity. The main advantage of the adopted declarative formalism is the fact that it facilitates the explanation of the domain, a point that is further discussed in the next chapter.

Chapter 5

Simulating and Explaining Supply Chain Operation

This chapter presents a framework for simulating and explaining the formal model that was introduced in Chapter 4. A rule-based approach is adopted for specifying the execution semantics of the formal model, based on which dynamic behaviours can be driven. This way the logic-based simulation and explanation of supply chain operation are enabled. Section 5.1 answers the question "How is the model simulated?" and presents the adopted framework and algorithm. Section 5.2 answers the question "How is the simulated model explained?" and distinguishes between two levels of explanation: low-level explanation of SC operation, which captures interdependencies in a detailed manner, and high-level explanation of problematic SC operation, which analyses the propagation of SC disruptions across the supply chain. Illustrative examples are used throughout this chapter to demonstrate how the suggested simulation and explanation framework help us understand SC operation dynamics.

5.1 Simulating SC Operation

As discussed in Chapter 3, simulation is a useful method for studying complex systems, such as supply chains. SC simulation allows experimentation with different SC operation scenarios, thus supporting SCM decision-making. In this section we

present the adopted framework for simulating SC-wide operation and we discuss aspects of an appropriately implemented simulation environment. Our aim is to fill the three gaps identified in existing SC simulation solutions, as presented in Chapter 3. In order to fill the first gap we adopt a knowledge-based approach, so as to enable the automated generation of explanations of SC operation dynamics. A mechanism for detecting SC disruptions is also provided, thus addressing the second gap. As far as the third gap is concerned, decision-making for agility purposes is simulated with the use of a reasoning engine.

5.1.1 Technical Design & Architecture

The purpose of simulating SC operation is twofold. Firstly, it provides an insight into SC operation dynamics with respect to SC members' behaviour and the resulting flow of products and information. Secondly, it allows the analysis of SC operation with respect to SC performance and problematic situations.

The main simulation *input* is the formal model of a supply chain. This includes the specification of the initial SC state through structural constructs and the specification of behavioural constructs for SC agents' layers. Additional input includes information on scheduled problematic situations (i.e. errors with items, process duration delays and lot size modifications) and on expected SC performance; this information is useful for simulating and detecting problematic SC operation, respectively. There are three categories of simulation *output*: (1) real-time SC operation, (2) measured SC performance and (3) detected problematic situations. Information on real-time SC operation involves SC members' behaviour (e.g. the firing of sourcing policies and the execution of making processes at a particular timepoint) and the flow of products and information (e.g. the movement of goods from a supplier to his customer). Measured SC performance and the detected problematic SC operation involve aspects discussed in Chapter 4.

The *architecture* of the simulation system is presented in Figure 5.1, where three main components can be seen: SC world, agents' resources and analysis tools. The SC world consists of a MAS of SC agents, the entities and information available and the SC events that occur. An SC agent consists of three layers, as discussed in Chapter 4: BRs, BPMs and communication capabilities. In order to exhibit dynamic

behaviour, an SC agent uses resources that drive SC simulation. The resources that are available to SC agents are: a workflow engine, a reasoning engine and a communication environment. As implied by the colours in Figure 5.1, these resources are linked to the SC agent's components: The workflow engine executes processes of an agent's BPM, and thus updates its workflow state. Similarly, the reasoning engine reads the SC agent's BRs and turns them into decisions towards actions for each state. The communication environment allows the exchange of messages within the SC through an appropriate infrastructure. Lastly, two tools analyse the overall SC simulation results: The SC performance calculator computes its performance, while the SC disruption detector identifies problematic SC operation.

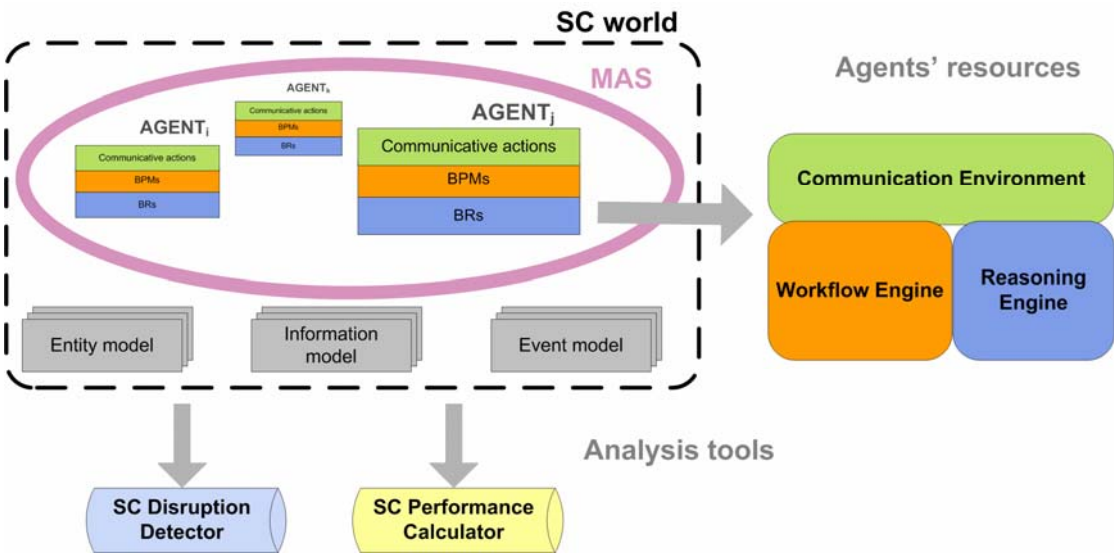

Figure 5.1: Simulation system architecture

Chapter 4 presented a declarative approach for modelling the SC world. Section 5.1.2 presents a logic-based approach for implementing the agents' resources and the analysis tools.

5.1.2 Logic-based Framework & Implementation

This section explains the main aspects of the implemented simulation system. As previously mentioned, a rule-based approach is adopted in order to support the automated explanation of SC operation dynamics. This approach is demonstrated here for the most important simulation procedures, and the simulation algorithm is

provided. More importantly, this section declaratively defines the execution semantics of the formal SC model, which was presented in Chapter 4. For this purpose we provide abstractions in the following form of production rules (Giarratano and Riley, 1998): perform operation: IF Conditions hold THEN enforce Effects. This means that an operation is performed if its conditions hold, and its performance brings about its effects. The syntax to be used is the following:

```
perform operation:
IF (  holds(Condition1)
      AND …
      AND holds(ConditionN)
      )
THEN ( enforce(Effect1),
       …
       enforce(EffectM)
      )
```

5.1.2.1 Workflow Engine

The workflow engine is used by SC agents to execute their BPMs. Its three main operations involve creating BPM instances, executing BP instances and executing junction instances. Before presenting these operations, it is worth clarifying the meaning of instances of BPMs, BPs and junctions. Suppose that we have a BPM that consists of one process and two junctions. During simulation we may have several occurrences of this BPM. This means that the process and each junction may be executed several times, possibly at different timepoints. In order to distinguish between the individual occurrences of BPMs, BPs and junctions, we refer to their instances. These instances follow the BPM, BP or junction specification that they correspond to, while being grounded to particular objects; for example a BP instance of process sup4_m11 (as presented in Section 4.3.4) has the same attributes as sup4_m11.

The rule for *process instance execution* is provided below. According to it, a process instance is executed if it has been reached, and its trigger conditions and preconditions hold. These conditions are explained later on in this section, but it is worth mentioning that whether a process instance is reached depends on the execution of junction instances within the BPM instance. Once a process instance

starts its execution, three effects take place: its execution completion is scheduled, its actions are scheduled for execution and any entities needed for its execution are assigned to it.

```
execute_process_instance(ProcessInst, TriggCond, Precond, Actions):
IF (  reached(ProcessInst)
      AND trigger_conditions_hold(TriggCond)
      AND preconditions_hold(Precond)
      )
THEN ( schedule_execution_completion(ProcessInst),
       schedule_actions_execution(Actions),
       assign_entities(ProcessInst)
      )
```

The preconditions of a process instance hold if each individual precondition within this set holds. We have specified rules for the holding of individual preconditions of all types discussed in Chapter 4, except for funds-related preconditions. Let us provide an example for illustration purposes. The precondition exist(entity_occ(EntityName), EntityAmount, EntityIds) holds at some SC agent if a set of entities EntityIds of amount EntityAmount and of type EntityName exist at the agent, and these entities are not assigned to any process instance execution.

The trigger conditions of a process instance hold if each individual trigger condition holds. We have specified rules for the satisfaction of individual trigger conditions of all types discussed in Chapter 4. For example, a trigger condition of the form exist(event_occ(EventName)) holds if an event of type EventName occurs at the SC agent.

As far as effects are concerned, the scheduling of the execution completion of the process instance takes into account the following: the duration defined in process/8 and any process duration delays occurring at run time. The actions execution of the process instance is scheduled for the time when its execution is completed. Lastly, any identified entities through the process instance's preconditions are assigned to the process instance for the duration of its execution.

We have implemented the execution of actions of all types discussed in Chapter 4, except for funds-related actions. The implementation for each action type is not further discussed here, but it is worth providing an illustrative example. The action

create_entity(ForAgentId, EntityName, EntityAmount, NewEntityIds) involves the creation of a set of entities NewEntityIds of amount EntityAmount, of entity type EntityName at the SC agent ForAgentId.

The assignment of entities to a process instance execution guarantees that the entities needed for its execution are not used by some other process instance execution. Once the process instance execution is completed, the assigned entities are released. For example, a machine that is assigned to some production activity A cannot be used by some other production activity B at the same time. But once the execution of A is completed, the machine can be assigned to the execution of B.

The rule for *junction instance execution* is provided below. According to it, a junction instance is executed if its type conditions hold (if its type constraints are satisfied). The execution semantics for each type of junction follow the FBPML specification (Chen-Burger et al. 2002), and we have specified appropriate rules for all junction types discussed in Chapter 4. For example, an and-joint junction instance has the following two conditions: (1) all process instances directly preceding it that have been triggered have also completed execution and (2) all junction instances directly preceding it have been executed. Once a junction instance is executed, the process instances directly following it are considered to be reached.

```
execute_junction_instance(JunctionInst, PostProcessInsts):
IF junction_type_satisfied(JunctionInst)
THEN reach(PostProcessInsts)
```

The rule for *BPM instance creation* is provided below. According to it, a BPM instance is created once the trigger conditions of its first process instance hold. Its effects involve the creation of all process instances and junction instances that it consists of.

```
create_BPM_instance(BPMInst):
IF  first_process_triggered(BPMInst)
THEN (create_process_instances(BPMInst),
      create_junction_instances(BPMInst)
      )
```

As mentioned in Section 4.3.2.2, previous work (Manataki, 2007) involved the design and implementation of a workflow engine for executing simple business process models within a single agent. Since this preliminary work was used as a basis

for developing a workflow engine for the PhD project, we should make clear what additional work was involved. The workflow engine presented in (Manataki, 2007) has been extended along four main lines: First, it has been enriched to be used in a multiagent setting, thus allowing the execution of BPMs of multiple agents. Second, the workflow engine developed in the context of (Manataki, 2007) allowed the execution of a single business process model for one run. The workflow engine presented in this section can execute several instances of various BPMs. This was achieved mainly thanks to the design and implementation of the above-presented procedure for BPM instance creation. Third, additional work involved assigning entities to process instance executions, as well as executing junctions that are preceded or followed by other junctions. To this end, the specification of process and junction instance execution semantics has been accordingly extended. Finally, the preliminary workflow engine has been enhanced to allow the execution of richer business process specifications, as discussed in Section 4.3.2.2.

5.1.2.2 Reasoning Engine

The reasoning engine is used by SC agents to execute their business rules. It enables the execution of three kinds of BRs, as presented in Chapter 4: (1) policy- and flexibility-BRs of ifthen(IFpart, THENpart) content form, (2) BRs for popular, customised policies, such as the (R,Q) policy and (3) process precondition BRs.

The rule for *executing a BR of* ifthen/2 *content form* is provided below. According to it, a BR of this type is executed if its IFpart is satisfied (remember that IFpart is a declarative expression of the conditions of the BR). Once such a BR is executed, the effects specified in the THENpart list are enforced.

```
execute_ifthenBR(BrId, IfPart, ThenPart):
IF br_condition_holds(IfPart) THEN enforce_effects(ThenPart)
```

The *execution of a BR for a customised policy* follows the execution semantics of that customised policy. For example, the execution conditions of an (R,Q) policy with content rq_policy(EntityName, R, Q) are satisfied if the on hand inventory level of product EntityName drops below R, and its effects involve the sourcing of amount Q.

The rule for *executing a BR of type process precondition* prescribes that such a BR is executed if the conditions expressed through its Content hold. Note that no

effects are enforced with its execution; nevertheless, the execution of such a BR can lead to the execution of the corresponding process instance, as it contributes to the satisfaction of its preconditions.

5.1.2.3 Communication Environment

The communication environment allows the agent to read and send messages to other SC members. The sending and receiving of inform-messages has been implemented. The rule for *reading messages* is provided below, and it assumes full trust between SC members. According to it, a message is read if it is received by the agent; note that a message is received by some SC agent if it is addressed to him or if it is broadcasted to all SC agents. The effect of reading an inform-message is that its content is added to the SC agent's knowledge base in the form of a fact.

```
read_message(MessageId, Content):
IF received(MessageId) THEN create_fact(Content)
```

Sending a message is considered to be an action additional to the ones mentioned in Section 5.1.2.1, executed through the communication environment. Once the sending of a message is invoked, a message is created (as specified in Chapter 4) and it is transferred to its recipients.

5.1.2.4 Performance Calculator

The SC performance calculator reads the simulation results for SC operation and computes the supply chain performance. We have implemented the calculation for following performance metrics that were discussed in Section 4.3.2.4: individual SC members' cost, on time rate and cycle times, as well as total SC cost. The formulae for calculating these metrics are presented in this section.

The formula for calculating an SC agent's cost is provided below, and it is based on the cost of all the process instances of the agent that have completed execution (hence $\mathrm{ProcessInst}_j$ is a process instance that has been executed by $\mathrm{Agent}i$). Note that the cost of a process instance is as specified at the corresponding process/8 declaration.

$$\mathrm{Cost}_{\mathrm{Agent}_i} = \sum \mathrm{Cost}_{\mathrm{ProcessInst}_j}$$

The total SC cost is calculated based on the cost of all SC members, as shown in the formula below.

$$\mathtt{TotalCost} = \sum \mathtt{Cost}_{\mathtt{Agent}_i}$$

The formula for calculating an SC agent's on time rate is provided below, and it is self-explanatory.

$$\mathtt{OnTimeRate}_{\mathtt{Agent}_i} = \frac{\mathtt{TotalNumberOfOrdersDeliveredOnTime}_{\mathtt{Agent}_i}}{\mathtt{TotalNumberOfOrdersDelivered}_{\mathtt{Agent}_i}}$$

An SC agent's cycle time for some SCOR operation for some product (e.g. for sourcing apples) is calculated based on the average execution time of the corresponding BPM (i.e. the sum of the average times of its processes). Note that a process's average time is based on the execution times of the corresponding process instances. In the formula provided below, $\mathtt{Process}_i$ is a process within the BPM of $\mathtt{Agent}i$ for SCOR operation $\mathtt{SCORtype}_s$ for product $\mathtt{Product}_p$.

$$\mathtt{CycleTime}_{\mathtt{Agent}_i,\mathtt{SCORtype}_s,\mathtt{Product}_p} = \sum \overline{\mathtt{Time}}_{\mathtt{Process}_j}$$

5.1.2.5 Disruption Detector

The SC disruption detector identifies problematic SC operation, as conceptualised and formalised in Chapter 4. The process of detecting certain types of problematic situations is simple, while for others it is more complex. Process duration delays and errors with items are fed into the simulation system, and hence detecting them is a matter of reading the simulation input. Unusual process instances executed are identified based on the characterisation of process types as unusual (e.g. if process `sup4_m11` is declared as unusual, then all its executed process instances are identified as unusual). The cancellations of order deliveries are typically communicated between SC members through messages, and thus they are tracked through the filtering of message content. Start and finish delays of process instances are detected by comparing the actual to the expected execution time start or completion, respectively. Big orders are detected upon their receipt by comparing the requested amount to its expected value.

Low SC performance is detected by comparing its actual to its expected or desired value. This means that there is a requirement for information on the expected or desired SC performance for the duration of the simulation.

We regard the detection of SC disruptions as an advantage of our approach compared to existing work in SC simulation, as presented in Section 3.1. Given that managing disruptions across the supply chain is becoming increasingly important in modern SCM practice (Melnyk et al. 2009), SC simulation should explicitly address SC disruptions.

5.1.2.6 Simulation Algorithm

So far we have presented the rule-based operations of different simulation modules. But when are these performed and how are they combined in order to simulate SC operation? This section answers precisely this question.

The top-level, centralised simulation algorithm is represented in Figure 5.2 in the form of an activity diagram. Two parts can be seen: a cyclic simulation part, which shows the sequence of simulations steps for each timepoint, and a part that corresponds to the steps at the end of simulation. Note that the user specifies the time period for which he/she wants to run the simulation (e.g. 23 timepoints), and hence simulation ends once this timepoint is reached.

The coloured steps in the diagram have already been discussed, and the colour of each step corresponds to the module in which it takes place (as in Figure 5.1). The white steps involve simulation aspects at the top level, such as initialising simulation based on the simulation input, and updating the time at the end of each simulation cycle. The "Enforce modifications" step enforces any modifications relevant to the current timepoint. These modifications are predefined at the simulation input and, as already mentioned in Section 5.1.1, they involve errors with items and lot size changes.

The names of several steps in Figure 5.2 end with "for all". This means executing the step for all SC agents, one after the other. Let us clarify what this means. As mentioned in Section 5.1.1, SC agents use three resources (i.e. a workflow engine, a reasoning engine and a communication environment) to drive their SC operational behaviour. Sections 5.1.2.1, 5.1.2.2 and 5.1.2.3 discussed when and how an individual SC agent uses these simulation modules. In order to simulate the operation of an entire supply chain, several SC agents need to use these resources.

Therefore, the steps with a name ending with “for all” refer to the case where all SC agents use these modules sequentially.

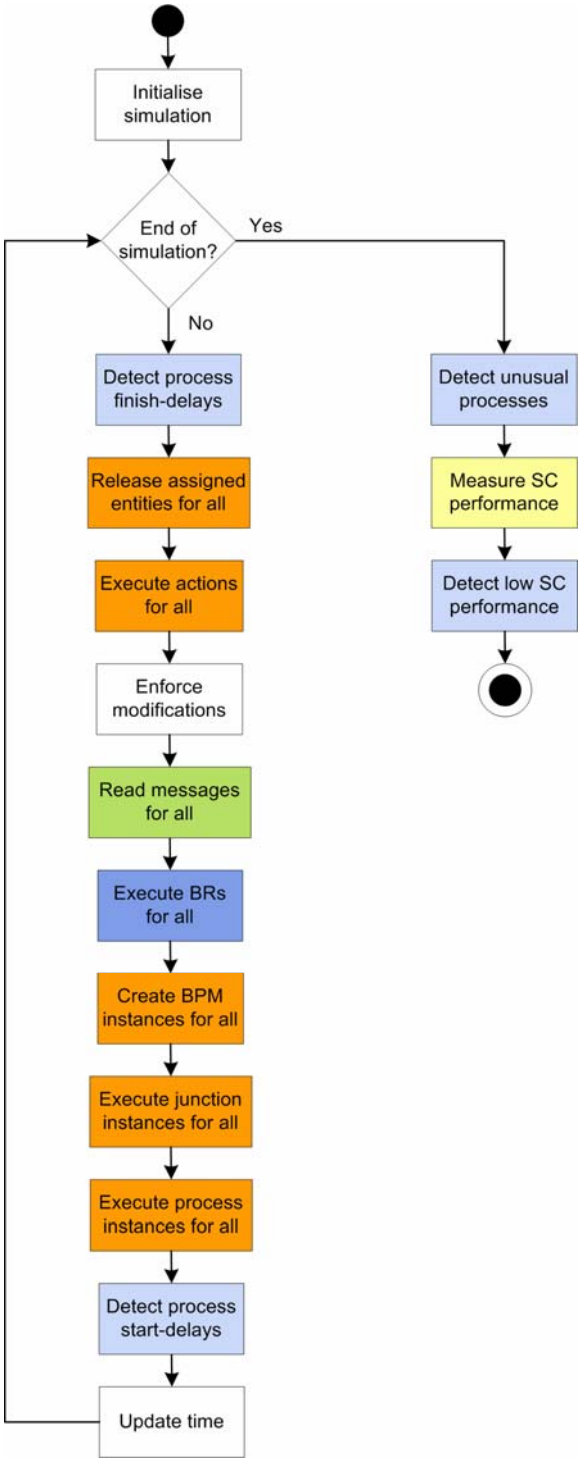

Figure 5.2: Simulation algorithm

The sequence of steps within the simulation algorithm of Figure 5.2 has been carefully decided so that SC agents exhibit smooth operational behaviour; for instance, assigned entities need to be released before new processes start executing, as their execution might require the availability of a previously assigned entity. It is worth mentioning that the timing of steps for measuring SC performance and detecting SC disruptions is flexible. This means that these steps could be implemented to take place at a different stage, while being equally correct or useful. Nevertheless, the simulation system implemented for the context of this thesis based on the algorithm of Figure 5.2 allows for correct and useful simulations of SC operation. This is demonstrated in the following section through an example.

5.1.3 Running Simulation Example

We will now illustrate the simulation approach discussed in this chapter for the operation of the example SC presented in Chapter 4. We first present the input for simulating this SC for 38 timepoints, then we discuss parts of the simulation output and, lastly, a walkthrough of the simulation algorithm is explained for a specific timepoint and SC member.

Throughout this section we will refer to particular SC constructs through their code names (i.e. their Ids). Many of the code names for the example SC are long and may seem hard to read; for this reason we will now explain how to read them. The general form of a process id is SCmember_SCORprocess, where SCmember is the code name of a member of the example SC, and SCORprocess is the code name of the SCOR-based process, as shown in Table 4.2. For instance, sup1_m16 is a process within Supplier1 of type M1.6. In the case of sourcing processes, the process id is of the form SCmember_SCORprocess_Product, thus also mentioning the sourced product involved. For instance, sup4_s11_p2 is Supplier4's sourcing process of type S1.1 for Product2. The general form of a junction id is j_SCmember_SCORProduct_Number, where SCORProduct refers to the SCOR operation type, as shown in Table 4.2, and the product involved. The ids of junctions of the same SCOR-based BPM, as visualised in Figure 4.4, differ only in the assigned Number. For instance, j_sup4_sp2_1 is a junction of Supplier4 within the BPM for sourcing Product2 and with the assigned number 1. Finally, process instances and junction instances have

code names of the form BPMid/ProcessId and BPMid/JunctionId, respectively. For example, process instance bpm-250/sup4_s11_p2 is an instance of process sup4_s11_p2, and junction instance bpm-250/j_sup4_sp2_1 is an instance of junction j_sup4_sp2_1. Note that any alternative code names could be used for the formalisation of the example SC, such as numbers; the above code names were chosen so that they carry some additional meaning.

The *input* for simulating the example SC includes the SC's formal model, as discussed presented in Chapter 4. It also includes information on the following scheduled problematic situations: 2 errors with items occur at Supplier1 (at timepoints 10 and 31), one error with items occurs at Supplier4 (at timepoint 16), 2 process duration delays occur at Supplier1 (the first of 1 timepoint for a sup1_m16 process instance at timepoint 7, and the second of 2 timepoints for a sup1_d13 process instance at timepoint 15), one process duration delay occurs at Transporter2 (of 1 timepoint for a trans2_d112 process instance at timepoint 8), Supplier4's sourcing P2 lot size is increased at timepoint 11 and Manufacturer's sourcing P4 lot size is increased at timepoint 28. The last type of simulation input involves the expected SC performance up to timepoint 38: total SC cost of 15460, on time rates of 1 and various values for cycle times (the full list is long and, therefore, it is not provided here).

As far as *outputs* are concerned, the user is firstly provided with information on real-time SC operation. This does not mean full information on each step of the simulation algorithm, but rather a selection of information that is interesting to SC managers. This includes information on process instances finishing execution and on the execution of their actions, on the receipt of messages by SC members, on the execution of BRs, as well as on the execution of process instances. Through this information the user can have an insight into not only the operational behaviour of SC members but also the flow of products and information involved. Figure 5.3 presents the output for SC operation in timepoint 4. Note that the output for timepoint 4 is fairly short, as not much is happening at the beginning of simulation; the output for a later timepoint (e.g. 28) is much longer. The user is also provided with information on SC performance, as shown in Figure 5.4. The last type of output involves problematic SC operation detected, and is shown in Figure 5.5. According

to it, several problematic situations of all types occur during simulation at several SC members. For instance, the execution of process instance bpm-745/sup4_m12 at Supplier4 starts late, order 1170 placed by Manufacturer to Supplier2 is unusually big and process instance bpm-251/sup4_s11u_p1 executed at Supplier4 is unusual, as it involves urgent sourcing. Supply chain performance is also detected to be lower in some cases: The total SC cost is higher than expected, the on time rates for Supplier1, Supplier2, Supplier4 and Manufacturer are lower than desired, and the cycle times for Supplier1's making and delivering P1 as well as Supplier2's delivering P2 are higher than expected.

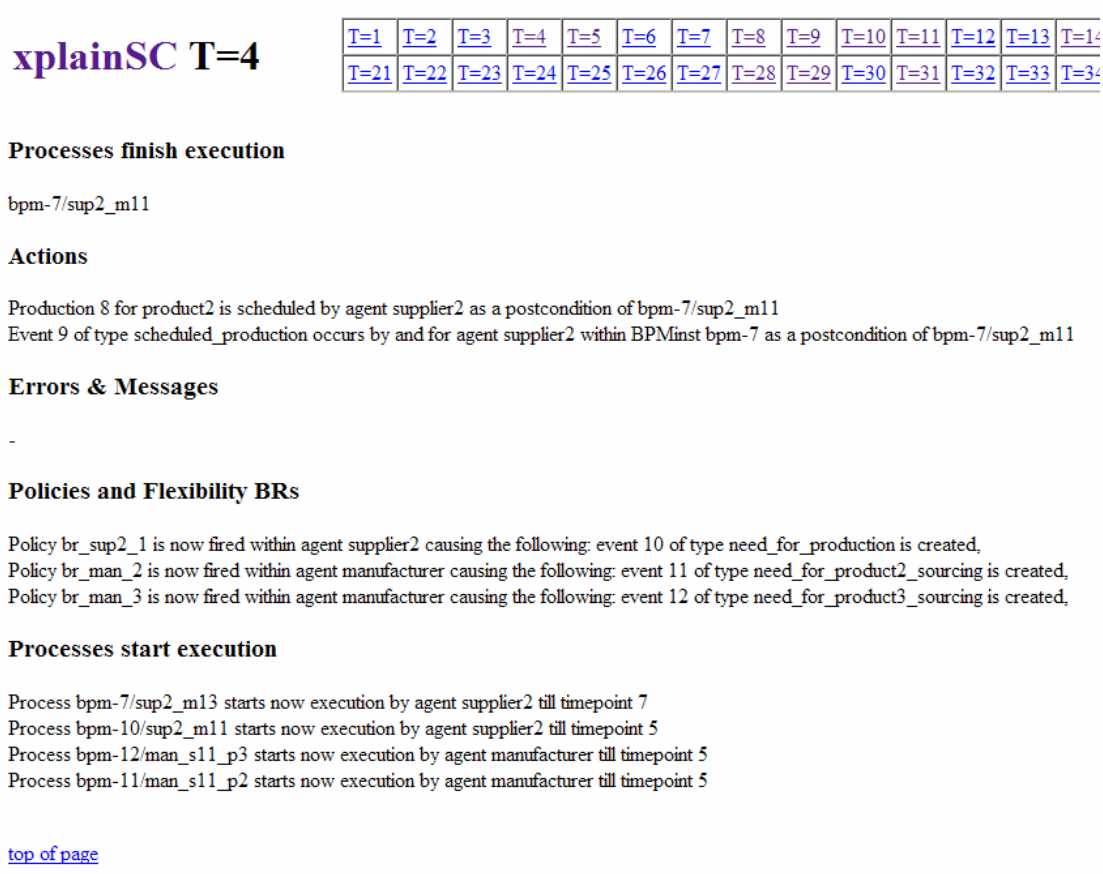

Figure 5.3: Simulation output for timepoint 4

Cost
The total cost for agent supplier1 is 980
The total cost for agent supplier2 is 3210
The total cost for agent transporter1 is 350
The total cost for agent transporter2 is 1400
The total cost for agent supplier5 is 600
The total cost for agent supplier4 is 2310
The total cost for agent supplier3 is 1330
The total cost for agent transporter3 is 350
The total cost for agent manufacturer is 5070
The total cost for agent retailer1 is 190
The total cost for agent retailer2 is 190
The total SC simulation cost is: 15980

On-time rates
The on time rate for agent supplier1 is 0.8571428571428571
The on time rate for agent supplier2 is 0.8928571428571429
The on time rate for agent supplier5 is 1.0
The on time rate for agent supplier4 is 0.8571428571428571
The on time rate for agent supplier3 is 1.0
The on time rate for agent manufacturer is 0.8823529411764706

Cycle times
The cycle time for agent supplier1 and make product1 is 3.1
The cycle time for agent supplier1 and deliver product1 is 4.2
The cycle time for agent supplier2 and make product2 is 5.0
The cycle time for agent supplier2 and deliver product2 is 5.035714285714286
The cycle time for agent supplier5 and deliver product1or2 is 1.0
The cycle time for agent supplier4 and source product1 is 4.0
The cycle time for agent supplier4 and source product2 is 4.0
The cycle time for agent supplier4 and source urgent_product1 is 2.0
The cycle time for agent supplier4 and source urgent_product2 is 2.0
The cycle time for agent supplier4 and make product4 is 5.0
The cycle time for agent supplier4 and deliver product4 is 5.0
The cycle time for agent supplier3 and deliver product3 is 3.0
The cycle time for agent manufacturer and source product4 is 3.0
The cycle time for agent manufacturer and source product2 is 3.0
The cycle time for agent manufacturer and source product3 is 3.0
The cycle time for agent manufacturer and make product5 is 6.0
The cycle time for agent manufacturer and deliver product5 is 5.0
The cycle time for agent retailer1 and source product5 is 2.0
The cycle time for agent retailer2 and source product5 is 2.0

Figure 5.4: SC performance output

We will now present a *walkthrough of the simulation algorithm* for Supplier4 at timepoint 11, clarifying what happens at each algorithmic step. Once the simulation cycle for timepoint 11 begins, process instance `bpm-211/sup4_s11_p1` completes its execution, as scheduled. No finish delay is detected for this execution (1st step of the simulation algorithm), and no entities are released (2nd step), as none were assigned to its execution. Six actions are executed by Supplier4 as an effect of this process instance execution (3rd step):

1. The ordered items' receipt time is scheduled for timepoint 15.
2. Order 235 for 6 product1 is placed to supplier1.
3. Event 236 of type `order_receipt` occurs at supplier1.

4. Message 237 with content order(235,supplier4,supplier1,supplier4,product1,6,15) is sent to supplier1.

5. Event 238 of type scheduled_order_receipt is scheduled for timepoint 15.

6. The on order inventory amount for product1 is increased by 6.

Process duration delay
Process bpm-35/sup1_m16 that starts execution by agent supplier1 at timepoint 7 has a duration delay of 1
Process bpm-110/trans2_d112 that starts now execution by agent transporter2 at timepoint 8 has a duration delay of 1
Process bpm-362/sup1_d13 that starts execution by agent supplier1 at timepoint 15 has duration delay of 2

Error with items
An error occurs at supplier1 at timepoint 10: 6 items of type product1 and status reserved/123 are no longer available
An error occurs at supplier4 at timepoint 16: 4 items of type product2 and status on_hand are no longer available
An error occurs at supplier1 at timepoint 31: 6 items of type product1 and status reserved/1162 are no longer available

Big order
Order 274 for 24 product2 placed by agent supplier4 is received by agent supplier2 at timepoint 13 through bpm-275/sup2_d12 and it is found to be bigger than usually
Order 1170 for 6 product4 placed by agent manufacturer is received by agent supplier4 at timepoint 30 through bpm-1171/sup4_d12 and it is found to be bigger than usually

Cancellation of order delivery
Supplier1 cancels the delivery of order 123 at timepoint 10
Supplier1 cancels the delivery of order 1162 at timepoint 31

Process start-delay
bpm-11/man_s12_p2, bpm-252/man_m12, bpm-275/sup2_d13, bpm-252/man_m13, bpm-331/sup2_d13, bpm-250/sup4_s12_p2, bpm-464/man_d13, bpm-303/man_s12_p2, bpm-335/sup4_s12_p1, bpm-636/man_m12, bpm-636/man_m13, bpm-425/ret1_s12, bpm-745/sup4_m12, bpm-745/sup4_m13, bpm-909/man_d13, bpm-945/sup4_d13, bpm-850/ret1_s12, bpm-914/man_s12_p4, bpm-1313/man_m12, bpm-1313/man_m13

Process finish-delay
bpm-35/sup1_m16, bpm-11/man_s14_p2, bpm-275/sup2_d111, bpm-331/sup2_d111, bpm-252/man_m16, bpm-362/sup1_d111, bpm-250/sup4_s14_p2, bpm-464/man_d112, bpm-335/sup4_s14_p1, bpm-636/man_m16, bpm-745/sup4_m16, bpm-945/sup4_d111, bpm-909/man_d112, bpm-914/man_s14_p4

Unusual processes execute
bpm-1361/sup4_s124u_p1, bpm-1387/sup5_d, bpm-1361/sup4_s11u_p1, bpm-469/sup4_s124u_p2, bpm-498/sup5_d, bpm-469/sup4_s11u_p2, bpm-251/sup4_s124u_p1, bpm-279/sup5_d, bpm-251/sup4_s11u_p1

Figure 5.5: Detected problematic situations

Supplier4's sourcing P2 lot size is increased from 16 to 24, as dictated by the simulation input (4th step). Two messages are read by Supplier4 (5th step): First, message 232 is read by Supplier4, and therefore the fact delivery(54,supplier4,supplier2,product2,16,[r-sup2-53,r-sup2-52,r-sup2-51,r-sup2-50,r-sup2-49,r-sup2-48,r-sup2-47,r-sup2-46,r-sup2-45,r-sup2-44,r-sup2-43,r-sup2-42,r-sup2-26,r-sup2-25,r-sup2-24,r-sup2-23]) is added to his knowledge base. This means that Supplier4 finds out about the delivery of order 54 for 16 items of Product2 (P2). Then, message 209 is read by Supplier4, and therefore the fact cancel_delivery(123,product1,6) is added to his knowledge base.

The execution of two BRs takes place at Supplier4 (6th step): First, policy br_sup4_2 is executed, causing the following: event 250 of type need_for_product2_sourcing is created. Second, flexibility BR br_sup4_urg2 is

executed, causing the following: event 251 of type need_for_product1_urgent_sourcing is created, the lot size for product1_urgent_lot_size is updated to 6, and the on order inventory amount for Product1 (P1) is decreased by 6.

Two BPM instances are created (7th step). The first involves sourcing P2, it is created due to event 250 and the created BPM instance includes process instances [bpm-250/sup4_s11_p2, bpm-250/sup4_s12_p2, bpm-250/sup4_s13_p2, bpm-250/sup4_s14_p2] and junction instances [bpm-250/j_sup4_sp2_0, bpm-250/j_sup4_sp2_1, bpm-250/j_sup4_sp2_2, bpm-250/j_sup4_sp2_3, bpm-250/j_sup4_sp2_4]. The second involves urgently sourcing P1, it is created due to event 251 and the created BPM instance includes process instances [bpm-251/sup4_s11u_p1, bpm-251/sup4_s124u_p1] and junction instances [bpm-251/j_sup4_sp1u_0, bpm-251/j_sup4_sp1u_1, bpm-251/j_sup4_sp1u_2].

Three junction instances are executed (8th step): bpm-211/j_sup4_sp1_1, bpm-250/j_sup4_sp2_0 and bpm-251/j_sup4_sp1u_0. The execution of three process instances begins till timepoint 12 (9th step): bpm-37/sup4_s12_p2, bpm-250/sup4_s11_p2 and bpm-251/sup4_s11u_p1. None of these are detected to start executing late (10th step).

It is interesting to note interdependencies between different aspects of Supplier4's SC operational behaviour during timepoint 11. For example, the receipt of message 209 on the cancellation of the delivery of order 123 leads to the execution of flexibility BR br_sup4_urg2, causing the occurrence of event 251. It is due to the occurrence of this event that a new BPM instance is created and process instance bpm-251/sup4_s11u_p1 is executed. One can, thus, conclude that the receipt of message 209 leads to the execution of bpm-251/sup4_s11u_p1.

Such interdependencies exist not only between aspects of a single SC member's behaviour, but also between the behaviours of several SC members. Explaining interdependencies in the case of complex supply chains over a long time horizon is not an easy task, especially when they consist of members with highly active SC operational behaviour. We argue that the study of such interdependencies, which constitute the dynamics of SC operation, is facilitated through the declarative formalisation of SC operation (presented in Chapter 4) and the adoption of a rule-based approach for implementing a simulation environment (presented so far in this

chapter). More interestingly, this study can be automated to provide users with answers to questions on SC operation and the dynamics involved. The rest of this chapter discusses how the adopted knowledge-based approach allows for the automated explanation of SC operation.

5.2 Explaining SC Operation

Supply chain operation dynamics can be analysed at two levels of detail: Firstly, detailed explanations can be provided about simulation results, so as to gain a deep understanding of interdependencies across the supply chain. Secondly, explanations at a higher level of detail can be generated on problematic SC operation; this way the propagation of SC disruptions can be analysed, and their effect on SC performance can be identified.

5.2.1 Low-level Explanation of SC Operation

In this section we present the adopted framework for generating detailed explanations of supply chain operation. This involves analysing SC operation dynamics with respect to the first four points of the research problem, as identified in Section 1.1. We discuss implementation decisions, and we provide an example of the use of the implemented explanation system.

5.2.1.1 Logic-based Framework

The explanation of SC operation at a low level involves explaining the simulation results with respect to four topics: (1) SC operational behaviour, (2) the state of the SC at a certain timepoint, (3) SC performance and (4) detected problematic SC operation. We believe that the most important type of question to ask on these topics is “why”. For example, one might want to find out why a particular process instance is executed at some SC member at some timepoint or why a specific product is available at some SC member at some timepoint. Similarly, the user might be interested to know how the on time rate for some SC member was calculated and why a finish delay was detected for a particular process instance.

Answering such questions is based on the rule-based execution semantics of the formal SC model. We believe that the choice of a declarative approach facilitates the explanation process. Figure 5.6 shows how SC operation can be explained given the production rule-based notation for describing execution semantics (as introduced in Section 5.1.2). According to it, an operation is performed because all its conditions hold, and some $Effect_i$ is enforced because the operation is performed. Let us clarify that for some $Condition_j$ to hold, the current SC state needs to be appropriate; note that the SC state is shaped by the enforcement of performed operations' effects.

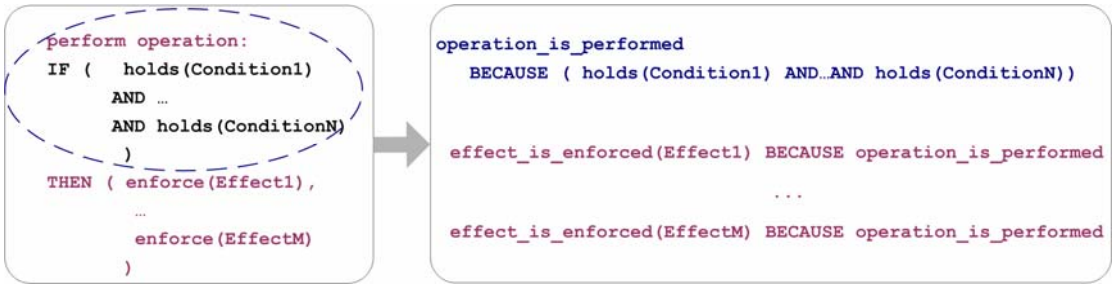

Figure 5.6: Explaining execution semantics

5.2.1.2 Implementation

The explanation of simulation results is implemented based on the rule-based execution semantics discussed in Section 5.1 and the mapping illustrated in Figure 5.6. The main idea involves keeping a simulation log that contains causal information, and deriving explanations based on this causal information. These two matters are further described in this section.

The simulation log is a report of interesting simulation events (here, by "events" we do not refer to SC events but to incidents that take place during simulation), such as the execution of process instances and the reading of messages by some SC member. This report does not only contain information on the simulation events that take place, but also on the reasons for which these take place. These reasons are deduced based on the formal execution semantics, as translated in Figure 5.6.

In our implementation, the simulation log contains information of the form fact(SimulationEvent, ListOfReasons, Timepoint). Three illustrative examples follow. According to the first fact, the entity r-man-462 of type Product5 is moved at timepoint 22 from Manufacturer to Retailer2; this happens because the action of moving such an entity is a post-condition of process instance bpm-515/man_d112,

which finishes its execution at timepoint 22. According to the second fact, the Manufacturer's on time rate is found to be 0.88 at timepoint 38 because Manufacturer delivers 17 orders in total, of which 15 are delivered on time. According to the third fact, a finish-delay is detected for process instance bpm-35/sup1_m16 because its execution is completed at timepoint 9 and not at timepoint 8, as scheduled.

```
fact(entity_is_moved(r-man-462,product5,retailer2,manufacturer),
  [post_condition(move_entity([r-man-462],product5,1,retailer2),
     bpm-515/man_d112),
   process_finishes_execution(bpm-515/man_d112,22)], 22).
fact(on_time_rate(manufacturer,0.88),
  [number_of_delivered_orders(manufacturer,17),
   number_of_orders_delivered_on_time(manufacturer,15)], 38).
fact(finish_delay_is_tracked(bpm-35/sup1_m16,supplier1,make,m16),
  [process_schedule_finish_time(bpm35/sup1_m16,8),
   process_actual_finish_time(bpm-35/sup1_m16,9)], 9).
```

Deriving explanations based on a simulation log that contains such causal information is a straightforward task. The process of explaining a simulation event SimulationEvent, for which there is relevant information of the form fact(SimulationEvent, ListOfReasons, Timepoint) in the simulation log, consists of retrieving its ListOfReasons.

It is interesting to note that each derived reason for some simulation event can be further explained following the same explanation process, thus generating a new set of reasons, which can in turn be explained, and so forth. This means, that a full explanation tree can be produced, if needed. We have implemented the explanation process in SICStus Prolog (Intelligent Systems Laboratory, 2003), which allows for use of recursion with ease, and facilitates the generation of such an explanation tree.

5.2.1.3 Running Example of Low-level Explanation

We will now demonstrate the type of low-level explanations on SC operation that can be generated following the framework described above. Consider the operation of the example SC that was formalised in Chapter 4 and simulated in Section 5.1.3.

Its simulation results can be explained with respect to all four topics mentioned in Section 5.2.1.1, but here we will focus on the first two topics.

The simulation output for timepoint 4, presented in Figure 5.3, includes that process instance `bpm-7/sup2_m13` starts executing at timepoint 4. There are three reasons for this fact: this process instance is reached at timepoint 4, and its trigger conditions and preconditions hold at this timepoint. The explanation derived following the adopted framework consists of these three reasons. Furthermore, an explanation can be generated at a larger depth, thus forming an explanation tree. We have generated such an explanation, a part of which is visualised in Figure 5.7. It is worth mentioning two points with respect to the generated explanation tree. Firstly, most tree nodes refer to SC operational behaviours (e.g. process instance execution) while others refer to the state of the supply chain (e.g. availability of a machine). Secondly, the generated explanation follows the execution semantics described in this chapter.

Explaining SC operation at a low level is particularly useful when one wishes to gain a deep and detailed understanding of overall SC operation. This way, for example, one can track which policy fired some making operation, and which specific product components were used for that making operation. Moreover, it can be discovered when these components arrived and from which supplier. The supplier's delivering operation can be further explained, if needed. However, explaining the propagation of problematic situations along the supply chain at such a low level can be hard to follow. Therefore, a higher level of explanation is needed for problematic SC operation. The process of generating such explanations is described in Section 5.2.2.

Figure 5.7: Part of an explanation tree for a process instance execution

5.2.2 High-level Explanation of Problematic SC Operation

In this section we present the adopted framework for generating high-level explanations of problematic supply chain operation. This includes analysing SC disruptions and identifying their effect on overall SC performance, thus covering the last two points of the research problem, as identified in Section 1.1. We detail implementation decisions, and we discuss illustrative examples to demonstrate the value of our approach.

5.2.2.1 Logic-based Framework

The explanation of problematic SC operation at a high level involves identifying causal relationships between detected disruptions. We identify the following five types of interesting questions with respect to causal relationships:

1. What is the reason for some SC operational problem?
2. What are the root causes of some SC operational problem?
3. What is the effect of some problematic situation?
4. What are all the effects (direct and indirect) of some problematic situation?
5. Given the occurrence of two SC operational problems, does one cause the other?

For example, one might want to find out why a particular order delivery was cancelled at some timepoint. Another interesting question could be about the root causes of high SC cost or about the set of effects of some error with items that occurred at some timepoint. Similarly, the user might be interested to know whether the duration delay of a specific process instance at some SC member led to low on time rate at some other SC member.

Answering such questions is based on the causal model that was presented in Section 4.3.3.2. We adopt a logic-based approach for specifying the explanation semantics of this causal model, as it is a natural choice when referring to causal relationships, and because it allows for reusability of the explanation process. The logical expression provided below defines when some SC operational problem B is the reason for some other SC operational problem A: if B is a possible reason for A

(according to the causal model), and both these problems hold and their specific occurrences are related. These points are further discussed in the following section.

$$possible\_reason(A, B) \wedge holds(A) \wedge holds(B) \wedge related(A, B) \rightarrow reason(A, B)$$

It is worth mentioning that the answering of all five identified types of questions can be based on the explanation semantics specified above, thus allowing for economy when implementing the explanation process. For instance, the notion of "effect" is the inverse of "reason", and hence if B is identified as a reason for A, then A is the effect of B. Furthermore, we can discover indirect reasons for some SC operational problem by recursively deriving reasons for it; this way, its root causes can also be identified. The transitive nature of the "reason" relationship also holds for its inverse, the "effect" relationship. Lastly, there is a causal path between two situations if one is the (direct or indirect) reason for the other. The following logical expressions describe these points. It is worth pointing out the generic nature of these logical expressions, a fact that is further discussed in Chapter 6.

$$reason(A, B) \rightarrow effect(B, A)$$
$$reason(A, B) \wedge reason(B, C) \rightarrow reason(A, C)$$
$$reason(A, B) \rightarrow causal\_path(A, B)$$

5.2.2.2 Implementation

The high-level explanation of problematic SC operation is implemented based on the causal model and its semantics discussed above, and it utilises the simulation log than contains causal information. The simulation log is useful for discovering whether some specific SC operational problem has occurred (i.e. for checking holds/1) and whether two particular SC operational problems are related (i.e. for checking related/2).

We have implemented holds/1 for all types of problematic situations in the causal model. Let us give two examples: First, a making operation finishes late if, according to the simulation log, a finish delay is detected for a process instance of SCOR type M1.6 (which is the last process within the make-BPM). Second, the products needed for some order delivery are not available on time if, according to the simulation log, a start delay is detected for a process instance of SCOR type D1.3 (which is the first

process within the deliver-BPM that requires the availability of products). The Prolog-based implementation for these examples is shown below:

```
holds(make_finish_delay(ProcInst)):-
      fact(finish_delay_is_tracked(ProcInst, _AgentId, make, m16),
        _Reasons, _T).

holds(needed_deliver_material_not_available(ProcInst)):-
      fact(start_delay_is_tracked(ProcInst, _AgentId, deliver, d13),
        _Reasons, _T).
```

Before discussing the implementation of related/2 it is worth clarifying the notion of relatedness between two specific problematic situations. In this context it is important to think of problematic situations as incidents that occur at some timepoint and involve some objects. Two specific problematic situations are typically related when they involve the same object. Let us provide a non domain-specific analogy to illustrate this point: Suppose that a flat's kitchen has a leak in January and that the same kitchen was flooded the preceding November. These two incidents are not related, as the water that leaks in the kitchen is not the same water of the flood (i.e. different "water-objects" are involved). This also makes sense given the temporal sequence of the two incidents. Such information is useful when one tries to discover whether the particular leak caused the flood: Even if a leak is a possible reason for a flood, and there was an occurrence of both a leak and a flood in the kitchen, that particular leak did not cause that particular flood, as the two incidents are not related.

Similarly, in the context of problematic SC operation, two situations are typically related when they involve the same object. For example, two delayed processes are related when they involve the same product or resource, or when they belong to the same BPM. A relevant example involves causal relationship 12, which states that "the needed products for delivering become available late because their making finishes late". The corresponding possible_reason/2 declaration, introduced in Section 4.3.3.2, is the following:

```
possible_reason(needed_deliver_material_not_available(D13ProcInst),
  make_finish_delay(MProcInst)).
```

The delayed make- and deliver-process instances for this causal relationship are related when they involve some common product. The implementation of related/2

for this causal relationship, shown in the Prolog code provided below, expresses precisely this idea. Note that the causal information within the simulation log is needed in order to check product_within_process_preconditions/2 and product_within_process_actions/2. It is worth mentioning that related/2 has been implemented for each possible_reason/2 declaration in the causal model. This implementation refers to the causal information within the simulation log.

```
related(needed_deliver_material_not_available(D13ProcInst),
  make_finish_delay(MProcInst)):-
      product_within_process_preconditions(EntityId, D13ProcInst),
      product_within_process_actions(EntityId, MProcInst).
```

Additional implementation involves non-causal, conceptual linking between problematic situations in the causal model; this is needed in order to support the recursive derivation of reasons for some situation. One example is the linking of late_delivered_order(AgentId, LateDeliveredOrder) with the corresponding deliver_finish_delay(DProcInst). Another example is the translation of a predicate with information on a list of items (e.g. unusual_processes(ListOfUnusualProcInst)) into several predicates, one for each individual item (e.g. unusual_process (UnusualProcInst1)).

Prolog was used for implementing the high-level explanation of problematic SC operation. Its recursive nature matches the transitive character of the "reason" and "effect" relationships, and enables the generation of explanation trees. This way the five identified questions on causal relationships are implemented in an elegant way. This matter is further discussed in Chapter 6.

5.2.2.3 Running Example of High-level Explanation

We will now demonstrate the high-level explanation derivation process for problematic SC operation. We refer to the example SC that was introduced in Chapter 4 and simulated in Section 5.1.3, and more specifically to its detected problematic operation. Explanations can be generated for all occurred problematic situations and in the form of all five types of questions identified in Section 5.2.2.1.

The first example involves identifying the direct reason for the start delay of Supplier4's process instance bpm-945/sup4_d13 of type "Reserve inventory", i.e. for

the fact that the needed products for delivering become available late. The reason identified by the system is the following: "There is a related make-finish-delay: bpm-745/sup4_m16". Figure 5.8 illustrates the two problematic situations in a graphical way, and one can see that the making delay at Supplier4 is propagated to his delivering operation. Note that the derivation of this explanation is based on causal relationship 12, the implementation of which was discussed in the previous section.

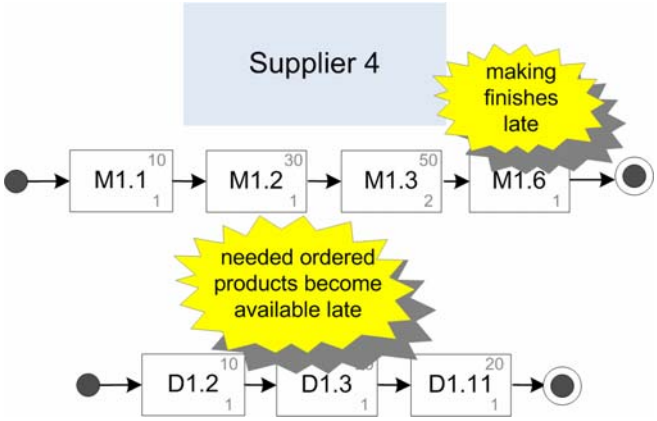

Figure 5.8: Propagation of delays at Suppllier4

The second example involves identifying the root causes for Manufacturer's low on time rate. Two root causes are identified by the system: (1) Transporter2's process instance bpm-110/trans2_d112 of type "Ship product" has a longer duration than scheduled and (2) order 274 placed by Supplier4 and received by Supplier2 (through his process instance bpm-275/sup2_d12) is unusually big. The generation of this explanation involves deriving reasons at a bigger depth, thus forming an explanation tree with several layers and branches. Figure 5.9 presents the system's output for this question, which consists of two parts: the second part provides the identified root causes, and the first part provides the explanation procedure for identifying the two root causes (hence corresponding to a textual form of the explanation tree). The propagation of the two identified problems along the supply chain is visualised in Figure 5.10 (the propagation is shown from left to right), where the yellow-marked problematic situations occur due to Transporter2's shipping delay and the green-marked problematic situations occur due to the big order placed by Supplier4. It is worth noting three points on this example: First, the provided explanation involves linking problematic situations and low SC performance. Second, there is a propagation of SC disruptions across several SC members (and tiers), and not just

within different operations of a single SC member. Third, one can see that Manufacturer's low on time rate was not his fault, but it was caused by disruptions at previous SC tiers, a case that is not unusual in real-world supply chains. We should emphasise that existing work in SC disruption analysis, as presented in Chapter 3, does not cover the identification of root causes of low SC performance. The example presented here demonstrates that our approach fills this gap for complex supply chains.

```
| ?- why_full(low_on_time_rate(manufacturer)).
------------------------------------------------------------------------------------------------------------
HOW TO READ THIS ANSWER:
- The answer has two parts:
The first is the chain of reasons that lead to the identified root causes, and it is provided right below.
The second is the identified root causes, and it is provided at the bottom in the block within asterisks.
- Which part to read?
If you want to find out the root causes of some problematic situation, then the second part suffices.
If you want to find out how these root causes were identified for the problematic situation (i.e. the full
 chain of reasons from the problematic situation to the identified root causes), then read the first part.
- How to read the first part of the answer?
The first part might have one or more blocks of text.
Each block contains several lines, where each line refers to a problematic situation that is a reason for
the preceding line.
In some cases, the last line of a block might mention several reasons for the preceding line. In that case
, additional blocks of text follow - each block explains each of those reasons.
------------------------------------------------------------------------------------------------------------
FIRST PART
The fact that the on time rate for manufacturer is low is due to the following:
The following received orders are delivered late by manufacturer : [908,463]
There are 2 reasons for this: the order 463 is delivered late by manufacturer and the order 908 is deliver
ed late by manufacturer
The fact that the order 463 is delivered late by manufacturer is due to the following:
There is a related deliver-finish-delay bpm-464/man_d112
The needed products are reserved late, i.e. a start-delay is tracked for bpm-464/man_d13
There is a related make-finish-delay bpm-252/man_m16
The needed material is issued late, i.e. a start-delay is tracked for bpm-252/man_m12
There is a related source-finish-delay bpm-11/man_s14_p2
The needed material is received late, i.e. a start-delay is tracked for bpm-11/man_s12_p2
There is a related deliver-duration-delay bpm-110/trans2_d112
Process bpm-110/trans2_d112 has a duration delay
The fact that the order 908 is delivered late by manufacturer is due to the following:
There is a related deliver-finish-delay bpm-909/man_d112
The needed products are reserved late, i.e. a start-delay is tracked for bpm-909/man_d13
There is a related make-finish-delay bpm-636/man_m16
The needed material is issued late, i.e. a start-delay is tracked for bpm-636/man_m12
There is a related source-finish-delay bpm-303/man_s14_p2
The needed material is received late, i.e. a start-delay is tracked for bpm-303/man_s12_p2
There is a related deliver-finish-delay bpm-331/sup2_d111
The needed products are reserved late, i.e. a start-delay is tracked for bpm-331/sup2_d13
There is a related earlier unusually big order at bpm-275/sup2_d12
The order dealt with by the deliver-BPM bpm-275 is unusually big

************************************************************************************************************
SECOND PART
Hence the root causes of the fact that the on time rate for manufacturer is low are the following 2:
process bpm-110/trans2_d112 has a longer duration than scheduled and the order received through the
delivering process bpm-275/sup2_d12 is unusually big
************************************************************************************************************
```

Figure 5.9: Explanation output for identifying the root causes of Manufacturer's low on time rate

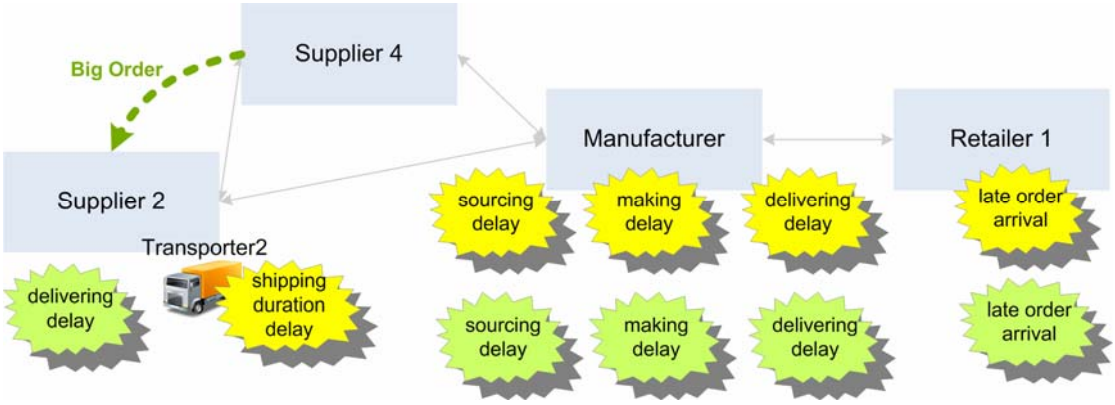

Figure 5.10: Propagation of delays across the SC, leading to low on time rate for Manufacturer

The third example involves identifying one direct effect of the error with P2 items that occurs at Supplier4 at timepoint 16. The effect identified by the system is that Supplier4 makes a flexibility decision on urgent sourcing for Product2 based on the execution of flexibility business rule br_sup4_urg1. Conceptually, this means that the error gives rise to urgent sourcing for Product2. This is shown graphically in Figure 5.11.

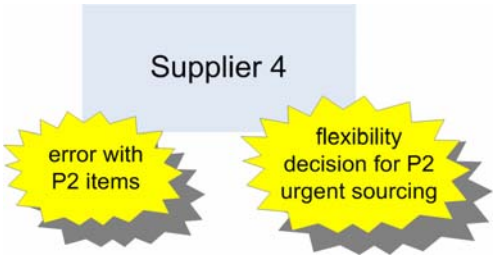

Figure 5.11: An error with items leads to a flexibility decision at Supplier4

The fourth example involves identifying all the effects (direct and indirect) of the error with P1 items that occurs at Supplier1 at timepoint 10. The most important effects identified by the system, visually represented in Figure 5.12, include the following: unusual processes execute at Supplier4 for urgent sourcing for Product1 (e.g. bpm-251/sup4_s11u_p1), delivered by Supplier5 through unusual process instance bpm-279/sup5_d, thus leading to high SC cost. Just like in the case of the second example, the generation of this explanation involves deriving effects at a bigger depth, thus forming an explanation tree. Figure 5.13 presents parts of the generated explanation tree.

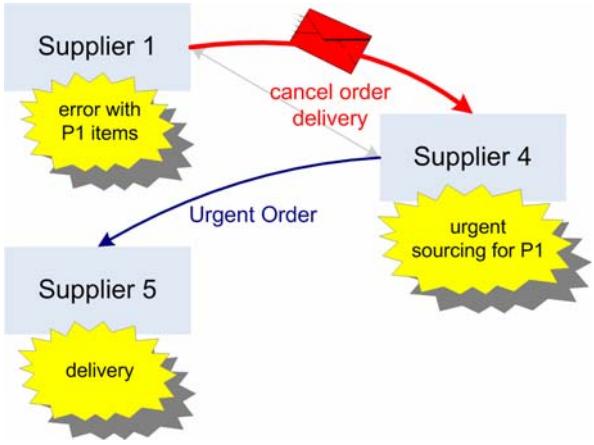

Figure 5.12: An error with items at Supplier1 leads to urgent sourcing at Supplier4

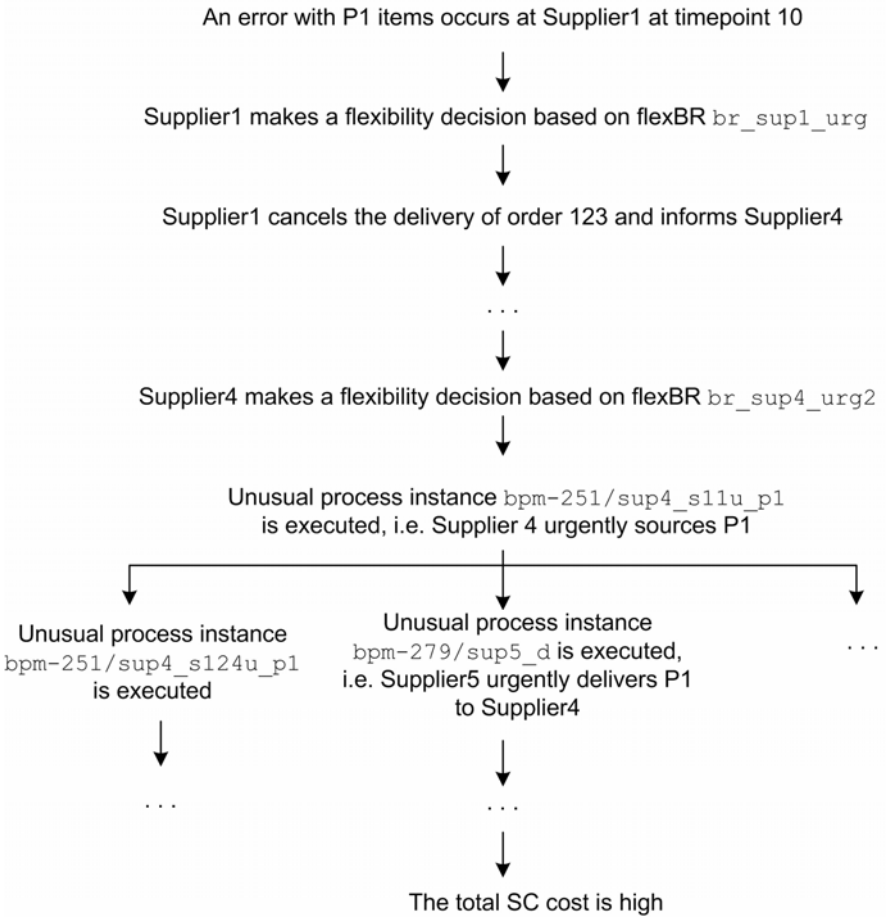

Figure 5.13: Explanation tree

The fifth example involves finding out whether the duration delay of Supplier1's process instance bpm-362/sup1_d13 of type "Reserve inventory" causes the make-finish-delay bpm-745/sup4_m16 at Supplier4. The system's answer to this question is

"yes", and the causal path between these two problematic situations is identified. Figure 5.14 shows the identified causal path in a graphical form.

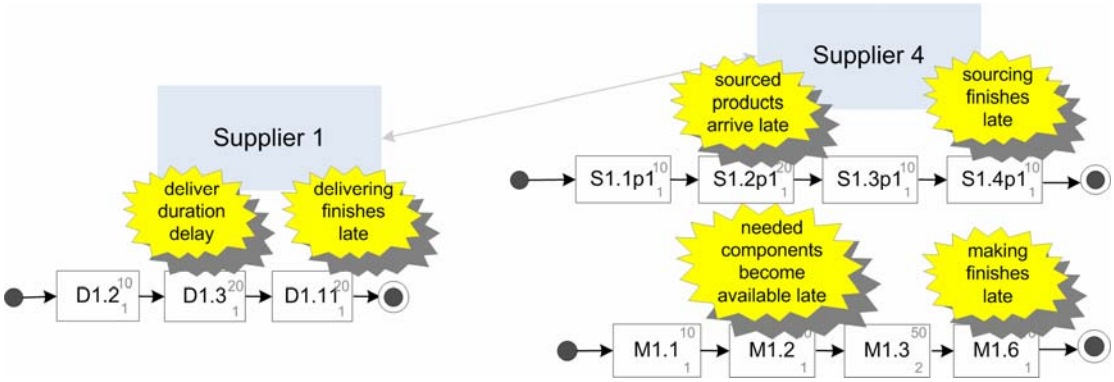

Figure 5.14: Causal path between deliver duration delay at Supplier1 and making finish delay at Supplier4

The five examples discussed above demonstrate how explanations on problematic SC operation are generated. We consider the generated explanations to be powerful with respect to three issues: First, the propagation of disruptions is tracked not only within an SC member's different SC operations (e.g. in the first example), but also across different SC members and across several SC tiers (e.g. in the second example). Second, the tracking of disruption propagation can involve disruptions of the same type (e.g. the fifth example involves only delays) or of different types (e.g. in the second example a received big order leads to a delivering delay). Third, low SC performance is explained and linked to occurred problematic situations (e.g. in the fourth example high total SC cost is linked to an error with items).

The presented explanation examples involved the problematic operation of a supply chain with complex operation dynamics, as discussed in Chapter 4. Moreover, the studied problematic SC operation included several problematic situations of all types, the relationships between which were non-obvious given the simulation results presented in Section 5.1.3. These two points make the task of explaining problematic SC operation for this scenario hard, yet feasible through our approach.

The generated explanations are useful, as they reveal how problematic situations are propagated across the SC, who is to blame for a certain problematic situation and what effects an SC disruption may have. This information is valuable, as it can be used to re-design aspects of the SC configuration and improve SC operation. For instance, alternative SC members may be sought to substitute a current SC member

whose problematic SC operation severely affects multiple SC tiers. Another example involves responding to delays by introducing time buffers or additional safety stock at some SC member.

5.3 Simulation and Explanation Summary

This chapter presented a logic-based approach for simulating and explaining SC operation. We declaratively specified the execution semantics of the formal model presented in Chapter 4. SC operation simulation was driven based on these semantics, and a suitable simulation algorithm was presented. We have accordingly implemented a simulation environment, the use of which was demonstrated through the simulation of the example SC. We believe that our framework allows for rich simulations, covering aspects such as SC members' behaviour and the resulting flows of products and information, measuring overall SC performance, and detecting problematic SC operation. The operation of entire supply chains can, thus, be studied, even when they have complex structures and dynamics.

In order to explain SC operation, the declarative execution semantics of the formal model were translated into grounded, low-level causal information, which was added to the simulation log during run-time. SC operation was explained at a low level based on this information, thus clarifying interdependencies within SC operation. Moreover, we presented a framework for explaining problematic SC operation at a higher level of detail. The generation of these explanations was based on the causal model that was formalised in Chapter 4, and it utilised the causal information within the simulation log. We have implemented an explanation system, and we have used it to answer five types of questions on the problematic SC operation of the example SC. In our opinion, the rule-based mechanism for generating these explanations is powerful, and the provided explanations are useful, as they can guide organised efforts towards SC improvement. This point is further discussed in the following chapter.

Chapter 6

Evaluation

In Chapter 1 we hypothesised that SCOlog generates explanations which provide useful insight into supply chain operation dynamics and employs a logic-based approach to the modelling and simulation of supply chain operation, allowing for maintainability and reusability. Chapter 4 presented a formal, declarative model of the domain, while Chapter 5 provided a rule-based reasoning mechanism for simulating and explaining the domain model. This chapter discusses the evaluation framework and results with respect to the research claims stated in Chapter 1. We, thus, answer the following questions: (1) Is this approach useful for understanding SC operation dynamics? (2) Does it improve the understanding of the domain for non-SCM experts? (3) Is it maintainable and reusable?

6.1 Evaluation Criteria & Framework

The criteria for evaluation have been defined based on the research claims outlined in Chapter 1. The thesis claims are as follows:

1. Automated explanation support is useful for the task of explaining supply chain operation dynamics, allowing for users' higher (a) time-efficiency, (b) correctness and (c) certainty about the explanations provided compared to the case where such support is not available.

2. The use of automated explanation support improves the performance of non-SCM experts, with respect to their (a) time-efficiency and (b) correctness when explaining SC operation dynamics. The correctness improvement is bigger compared to the case where no automated explanation support is available, without loss of time-efficiency. This suggests that the use of automated explanation support improves the understanding of the domain for non-SCM experts.
3. A logic-based approach for modelling, simulating and explaining SC operation scenarios allows for maintainability and reusability with respect to (a) the specified SC operation input models, (b) the developed simulation system and (c) the developed explanation system.

The evaluation criteria that correspond to each research claim are the following: (1) approach usefulness, with three sub-criteria of users' efficiency, correctness and certainty, (2) users' performance improvement, with two sub-criteria of correctness improvement and efficiency improvement and (3) maintainability and reusability.

Deciding on the evaluation method for each of these claims and criteria is highly influenced by their nature. A user-based empirical evaluation is appropriate for the first two research claims, as they involve the understanding of the domain for human users. An example-driven analytical approach is adopted for evaluating the third claim, as it involves qualities of the developed computational and reasoning model.

6.2 Empirical Evaluation Design

6.2.1 Scenarios

One way of empirically evaluating the first two research claims is through real-world SC scenarios. This involves representing the operation of a real SC at a satisfying level of detail, as well as capturing information on SC performance and occurring problematic situations. Unfortunately, getting access to such – often sensitive – information is extremely hard. For this reason, typical SC scenarios have been developed that cover three requirements: First, they are representative of the SCM domain; second, they are complex enough to demonstrate the system's explanatory

power; third, their level of difficulty is appropriate, meaning that their dynamics can be understood and explained by domain experts when no system support is available.

The developed SC scenarios involve the operation of the example supply chain presented in this thesis (remember that its formalisation was introduced in Chapter 4, while its simulation and explanation was described in Chapter 5). The operation of this supply chain has been extensively discussed in these chapters, and therefore we will not discuss it any further here. Nevertheless, it is worth repeating a few basic points: The example SC consists of eight main SC members across four tiers, and it involves the flow of five types of products. Apart from standard SC decisions, flexibility decisions are also made during its operation. Problematic situations that occur during operation involve delays, errors with items, big orders, cancellations of order deliveries and unusual processes; these lead to low SC performance.

Three simulation scenarios have been developed for the example SC: Scenario1 (which was simulated and explained in Chapter 5), Scenario2 and Scenario3. Scenario1 involves an SC simulation for 38 timepoints, Scenario2 is run for 42 timepoints, while Scenario3 is run for 38 timepoints. Several problematic situations of all types occur at all SC members (apart from Supplier3 and Retailer2) in the scenarios; what differs between the three scenarios is the timing, the number, the location and the propagation degree of individual problematic situations. It is worth mentioning that the task of explaining the problematic SC operation of the three scenarios is non-trivial. This is mainly due to the big number and wide range of problematic situations that occur, as well as their varying propagation degree (i.e. some problematic situations are not propagated, while others are propagated from the first to the last tiers of the supply chain, affecting overall SC performance). The simulation and explanation of Scenario1 in Chapter 5 demonstrated this fact.

The three scenarios were carefully designed so that their operation dynamics complexity is of the same scale. Scenario1 and Scenario3 are simulated for the same time period, and they involve the same number of problematic situations of each type (e.g. three processes have a duration delay in both scenarios). Furthermore, there is a direct mapping between the propagation of individual problematic situations in the two scenarios (e.g. the propagation of a process duration delay at Transporter2 in Scenario1 is similar to the propagation of a process duration delay at Supplier2 in

Scenario3). As far as Scenario1 and Scenario2 are concerned, Scenario2 has a longer simulation horizon but it involves a smaller total number of problematic situations; this way a balance is established between the two scenarios. Furthermore, there is a direct mapping between the propagation of individual problematic situations in Scenario1 and Scenario2 (e.g. both scenarios involve the occurrence of an error with items at Supplier4, with similar effects in the two scenarios).

The supply chain and the scenarios described here are appropriate for evaluation purposes, as they satisfy the three requirements discussed at the beginning of this section. Firstly, the SC is representative of real-world supply chains, consisting of several tiers and involving the flow of different products. Secondly, the complexity requirement is satisfied not only through the structure of the supply chain, but also given the product-based interdependencies between the SC members (this issue was also discussed in Chapter 4) and the simulation time horizon of the scenarios. Another complexity factor involves the number, type and propagation degree of problematic situations. Thirdly, the pragmatism requirement is satisfied, as domain experts can successfully explain the dynamics of the three scenarios even when no system support is available. Finding the right balance between the last two requirements was achieved through pilot runs with users.

6.2.2 Tasks & Subjects

Experiment participants were asked to answer questions on SC operation dynamics for one or more of the above-described scenarios. More specifically, the questions involve causal relationships between arisen problematic situations in a scenario, and they were of the following three types discussed in Section 5.2.2.1: identifying root causes, identifying all effects, and identifying the causal path between two problematic situations. The questions asked for each scenario cover different aspects of SC operation dynamics and different problematic situations, thus avoiding any learning effects in the participants. They also have equivalent levels of difficulty (e.g. explaining the propagation of one problematic situation across two SC tiers is regarded as equivalent to explaining the propagation of two problematic situations across one SC tier). Therefore, we consider the questions for each scenario as equally important and informative with respect to the explanation performance of

participants. The experiment tasks were carefully designed, so that a consistent level of difficulty is achieved between questions not only within the same scenario, but also across different scenarios. This is particularly important for the evaluation of the second research claim, and it is further discussed in Section 6.4.

The answers were provided by the experiment participants in text. The nature of most questions was open (e.g. “The on-time rate of Manufacturer is lower than expected. Identify all root causes of this situation.”). The only exception were questions on the causal path between two problematic situations, consisting of two parts: (1) a closed yes/no question (i.e. “Does situation X cause situation Y?”), and, in the case of a yes answer, (2) an open question on the causal path (i.e. “Provide the causal path between situations X and Y.”). Participants were given at most 6 minutes to answer each question. The time constraint was set for practical reasons, so that experiments do not have duration longer than one and a half hour. The 6 minute limit was set after pilot runs, and it allows for correct answering in the case where no system support is available.

A total number of 28 people have participated in the experiments for the empirical evaluation. We distinguish between two classes of participants with respect to their expertise: SCM experts and business experts. SCM experts have a deep knowledge of SCM subjects, either from a theoretical or a practical point of view. Examples of SCM experts include SCM scholars (from PhD students and young lecturers to professors with research experience of more than twenty years), SCM practitioners (a logistics manager and a product procurement manager), as well as students and holders of an MSc in Logistics and SCM. The class of business experts includes scholars and practitioners in the wider area of business and management, who do not have SCM expertise. Hence, all subjects with business expertise were carefully selected so that they do not have advanced knowledge of SCM; some basic knowledge of the domain was, however, allowed. Examples of business experts include people working in industry, such as Marketing and Communications managers, as well as students and holders of an undergraduate or postgraduate degree in an area relevant to Business Administration and Management.

6.3 Usefulness of the Approach

This experiment aims to show that automated explanation support is useful for the task of explaining supply chain operation dynamics. The experiment was designed to facilitate the comparison between the performance of subjects in two cases: when automated explanation support is provided and when such support is not available. Tests of users' efficiency, correctness and certainty were performed to evaluate this claim.

6.3.1 Experimental Setup

Two SC scenarios were used for this experiment, Scenario1 and Scenario2. As discussed in Section 6.2.1, these scenarios involve several problematic situations of all types, while maintaining equivalent levels of difficulty when it comes to explaining their dynamics. Four questions were asked for Scenario1 (as presented in Appendix A) and three for Scenario2, covering the three question types mentioned in Section 6.2.2. However, the questions for each scenario focus on different and independent groups of problematic situations: Questions for Scenario1 focus on delays and big orders that may cause high cycle times and low on time rates, while questions for Scenario2 focus on errors with items and cancellations of order deliveries, which may cause the execution of unusual processes and high SC cost. This differentiation was made in order to avoid any learning effects for the participants, and it will be further discussed at the end of this section.

In this experiment, subjects were asked to answer scenario questions with or without automated explanation support. In the case where automated explanation support was available, subjects did not interact directly with the developed system for asking questions – this way, and given the absence of a graphical user interface for the developed system, syntactic mistakes were avoided. Instead, participants specified the type of question they wanted to make (e.g. find_all_effects) and the subject of the question (e.g. low on time rate), and the experiment conductor typed the corresponding Prolog query. We should also note that at the beginning of the experiment subjects were provided with information on possible types of questions that the system can answer.

When automated explanation support was provided, the process of answering a question was as follows: Firstly, experiment participants specified the type and subject of the question they wanted to make. Secondly, the experiment conductor typed the corresponding Prolog query and run it. Thirdly, subjects were provided with the system's answer, which they could copy and paste in the questionnaire, alter or completely ignore. If interested, subjects could be provided with information on all problematic situations that occurred in that scenario.

When no automated explanation support was provided, subjects answered questions based on the corresponding scenario's simulation results. The simulation results for the experiment's scenarios were presented to the subjects not through the SICStus Prolog's interpreter window, but instead in appropriately designed HTML files. This way, the simulation output was more user-friendly, and the subjects' navigation through the simulation results was facilitated.

For each subject, three variables were measured for the answering of each scenario question: time to answer the question, correctness of the provided answer and subjective certainty about the answer. Time was measured on the spot by the experiment conductor and the measurement unit is seconds. The correctness of each answer was graded from 0 to 10, by comparing it to the ground truth that was established during the experimental design. A carefully designed marking scheme was devised for this task, rewarding correct but incomplete answers, and deducting marks for incorrect answer sub-points, when needed. The certainty about the given answer was graded from 0 to 4, and it was specified by the subject in the following way: Subjects were told they were participating in a betting game, in which they had a number of betting chips available, and they could bet from 0 up to 4 for each answer. Since the certainty of subjects about the given answer was self-assessed, we regard the measured certainty to be subjective certainty; for reasons of simplicity, however, 'subjective certainty' is called 'certainty' throughout this thesis. An average value was calculated for each measured variable over all questions for the same scenario answered by each subject. This way, three metrics were available for each participant's explanation performance for each scenario: his average time to answer the scenario's questions, his average correctness and his average certainty.

	Group1	Group2
Scenario1	Y	N
Scenario2	N	Y

Table 6.1: Availability of automated explanation support per group and scenario

The number of subjects in this experiment was 20, consisting of 14 SCM experts and 6 business experts. These 20 subjects were split in two groups, Group1 and Group2, of equal sizes and proportions with respect to expertise. The rationale behind distinguishing between two groups lies in the choice of a between-group comparison of subjects' performance. A within-group experiment setup was rejected, as it would severely suffer from a learning effect. All subjects answered all questions for the two scenarios, some with and some without automated explanation support. The availability of automated explanation support per group and scenario is shown in Table 6.1, according to which members of Group1 were provided with automated explanation support for Scenario1, but not for Scenario2. Similarly, members of Group2 were provided with automated explanation support for Scenario2, but not for Scenario1. This way, and given that the tasks for the two scenarios have similar difficulty but involve different types of problematic situations, the sample sizes for this experiment were doubled. It is worth clarifying that in this experimental design there is no learning involved for subjects of any group, as the questions of the two scenarios focused on different and independent sets of problems. Let us illustrate this with an example: Consider a participant of Group1 who first answered questions for Scenario1 with the use of automated explanation; these questions involved delays and big orders that caused high cycle times and low on time rates. When this participant was then asked to answer questions for Scenario2, he couldn't use any of the knowledge he gained through Scenario1, as the questions for Scenario2 involved a different set of problems, mainly caused by errors with items. The fact that no such learning takes place is particularly important for the independence between the two samples used in our statistical tests.

6.3.2 Results

Tables 6.2, 6.3 and 6.4 contain the average time, correctness and certainty measurements, respectively, for all experiment participants in two cases: where explanation support is available and where it isn't. It is worth reminding the reader that time values may range from 0 to 360 seconds (as the maximum allowed time was 6 minutes), correctness values may range from 0 to 10, while certainty values may range from 0 to 4. The table columns illustrate the choice of a between-group comparison, while their rows reveal the sample sizes for the two cases compared. Based on these values, statistical tests were performed to evaluate the efficiency, correctness and certainty of the task of explaining SC operation dynamics. These tests are discussed in the following three sections.

	Time with automated explanation support	**Time without automated explanation support**
Scenario 1	201.5	351.25
	149.5	356.5
	222.75	326.5
	280	283.25
	241.5	360
	241	360
	242.75	344.5
	294.5	360
	262.5	315
	210.5	337.75
Scenrio2	245	295.33
	266	303
	132	282.67
	85.33	246.33
	229	346
	271.67	360
	230.67	340
	122	297.33
	129	332.67
	211	312.67
Average	213.4	325.54

Table 6.2: Time for providing answers with and without automated explanation support

	Correctness with automated explanation support	Correctness without automated explanation support
Scenario 1	10	4.25
	10	4.25
	7.5	4.5
	9.5	3
	10	4.25
	10	0.25
	10	3.6
	6	2.25
	10	2.25
	8.75	0.75
Scenrio2	10	1
	6.67	1.33
	10	1.33
	10	1.33
	10	0.33
	10	2.33
	9	4
	10	1.33
	10	0
	10	2
Average	9.37	2.22

Table 6.3: Correctness of answers with and without automated explanation support

	Certainty with automated explanation support	Certainty without automated explanation support
Scenario 1	4	2.25
	3	0.5
	2.75	3.5
	3.25	2.25
	2.75	3.25
	2.5	1
	3.75	1.75
	2.25	0.75
	2.25	2.25
	3.75	0.5
Scenrio2	3.33	1
	2.67	0
	4	3
	4	3.33
	3	2.67
	2.33	1
	2.33	2
	4	2.33
	4	1
	3	3
Average	3.16	1.87

Table 6.4: Certainty about answers with and without automated explanation support

6.3.3 Test of Efficiency

Statistical hypothesis testing using the t-distribution was conducted to evaluate research claim 1a) on time-efficiency, as presented in Section 6.1. The rationale behind the adoption of the t-test is explained later on in this section. The hypothesis, null hypothesis, independent and dependent variables for this test are provided below.

- Hypothesis:

  Explaining SC operation dynamics with the use of the explanation system takes less time compared to the case where the explanation system is not available (i.e. when explaining is based on simulation outputs).

- Null Hypothesis:

There is no difference in the time taken to explain SC operation dynamics with and without the use of the explanation system.

- Independent variables:

  – Data: 2 scenarios of complex and problematic SC operation (Scenario1, Scenario2)

  – Tasks:

  T1: answer questions on the dynamics of an SC scenario with the use of the explanation system

  T2: answer questions on the dynamics of an SC scenario without the use of the explanation system

  – Subjects: 20 subjects for T1 and 20 subjects for T2, of which 14 SCM experts and 6 business experts

- Dependent variables:

  – Time taken to perform task T1 versus time taken to perform task T2

A one-tailed t-test was performed to determine the t value and its corresponding p value in order to accept or reject the null hypothesis. A significance level of $p < 0.05$ was regarded as an acceptable condition to reject the null hypothesis. The t-test was chosen based on the sizes of the two samples, and considering that the population variances are unknown (Anderson et al. 2004). Given the choice of a between-group comparison of time to perform T1 and T2, the two sample independent t-test was performed. Note that the two samples are independent for the reasons explained in Section 6.3.1. The t value is given by the following equation:

$$t = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

where $\overline{x_i}$ is the mean of each sample, s_i is the standard deviation of each sample and n_i is the size of each sample.

The data of Table 6.2 was used for this test, and the calculated t-value was found to be 7.439. This value corresponds to a significance level of $p=1.686 \times 10^{-8}$, which is much smaller than the significance level of 0.05. This means that the null hypothesis can be rejected. Hence we can conclude that the efficiency of explaining SC

operation dynamics with the use of the explanation system is significantly higher compared to the case of no explanation system availability.

6.3.4 Test of Correctness

A similar statistical hypothesis test using the t-distribution was conducted to evaluate research claim 1b) on the correctness of provided explanations, as presented in Section 6.1. The rationale behind the choice of t-test is similar to the one for the test of efficiency, discussed in Section 6.3.3. The hypothesis, null hypothesis, independent and dependent variables for this test are provided below.

- Hypothesis:
  The correctness of the explanations of SC operation dynamics when using the explanation system is higher compared to the case of no explanation system use.
- Null Hypothesis:
  There is no difference in the correctness of the explanations of SC operation dynamics that are provided with and without the use of the explanation system.
- Independent variables:
  – Data: 2 scenarios of complex and problematic SC operation (Scenario1, Scenario2)
  – Tasks:
  T1: answer questions on the dynamics of an SC scenario with the use of the explanation system
  T2: answer questions on the dynamics of an SC scenario without the use of the explanation system
  – Subjects: 20 subjects for T1 and 20 subjects for T2, of which 14 SCM experts and 6 business experts
- Dependent variables:
  – Correctness of explanations for task T1 versus correctness of explanations for task T2

Ground truth: determined by experiment designer

Similarly to the test presented in Section 6.3.3, a one-tailed two sample independent t-test was performed to determine the t value and its corresponding p value in order to accept or reject the null hypothesis. A significance level of $p < 0.05$ was regarded as an acceptable condition to reject the null hypothesis.

The data of Table 6.3 was used for this test, and the calculated t-value was found to be 16.581. This value corresponds to a significance level of $p=5.145x10^{-19}$, which is much smaller than the significance level of 0.05. This means that the null hypothesis can be rejected. Hence we can conclude that the correctness of explanations of SC operation dynamics that are provided with the use of the explanation system is significantly higher compared to the case of no explanation system use.

6.3.5 Test of Certainty

A similar statistical hypothesis testing using the t-distribution was conducted to evaluate research claim 1c) on the subjective certainty about provided explanations, as presented in Section 6.1. We chose the t-test for the same reasons that were discussed in Section 6.3.3. The hypothesis, null hypothesis, independent and dependent variables for this test are provided below.

- Hypothesis:

  The certainty about the explanations of SC operation dynamics that are provided with the use of the explanation system is higher compared to the case of no explanation system use.

- Null Hypothesis:

  There is no difference in the certainty about the explanations of SC operation dynamics that are provided with and without the use of the explanation system.

- Independent variables:

  – Data: 2 scenarios of complex and problematic SC operation (Scenario1, Scenario2)

  – Tasks:

T1: answer questions on the dynamics of an SC scenario with the use of the explanation system

T2: answer questions on the dynamics of an SC scenario without the use of the explanation system

– Subjects: 20 subjects for T1 and 20 subjects for T2, of which 14 SCM experts and 6 business experts

- Dependent variables:

  – Certainty about the explanations for task T1 versus certainty about the explanations for task T2

Similarly to the tests presented in the previous two sections, a one-tailed two sample independent t-test was performed to determine the t value and its corresponding p value in order to accept or reject the null hypothesis. A significance level of $p < 0.05$ was regarded as an acceptable condition to reject the null hypothesis.

The data of Table 6.4 was used for this test, and the calculated t-value was found to be 4.517. This value corresponds to a significance level of $p=4.021 \times 10^{-5}$, which is much smaller than the significance level of 0.05. The null hypothesis can, thus, be rejected, and we can conclude that the certainty about explanations of SC operation dynamics that are provided when using the explanation system is significantly higher compared to the case of no explanation system use.

6.3.6 Discussion

An experiment with participants of SCM and business expertise was conducted to empirically evaluate the usefulness of automated explanation support provided by SCOlog. Experiment participants were asked to answer questions on complex and problematic SC operation scenarios with or without the use of the explanation system. Time, correctness and certainty measurements were taken for the users' answering process. The collected data was used for three statistical hypothesis tests addressing efficiency, correctness and certainty. Based on these tests we concluded the following:

- The users' efficiency of explaining SC operation dynamics with the use of the explanation system is significantly higher compared to the case of no explanation system use.
- The users' correctness of explanations of SC operation dynamics that are provided when using the explanation system is significantly higher compared to the case of no explanation system use.
- The users' certainty about explanations of SC operation dynamics that are provided when using the explanation system is significantly higher compared to the case of no explanation system use.

It is also worth discussing the magnitude of the difference of means between participants' performance in the two cases (i.e. with vs. without the use of the explanation system). As it can be seen from Table 6.2, the average time for answering questions without the use of the explanation system was more than 150% longer than in the case where the explanation system was available. According to Table 6.3, the average correctness of answers provided with the use of the explanation system was more than 400% higher than in the case where the explanation system was not available. Given the data shown in Table 6.4, the average certainty of participants about the answers provided with the use of the explanation system was more than 160% higher than in the case where the explanation system was not available.

The above points, along with the results of the statistical hypothesis tests, demonstrate that automated explanation support is useful for the task of explaining supply chain operation dynamics. It is useful, as it allows SCM and business experts to quickly and correctly explain SC operation dynamics, while feeling confident about their understanding.

Additionally, experiment participants were asked to directly assess the usefulness of the automatically generated explanations by providing a grade from 1 to 5. As shown in Appendix B, 1 corresponds to not useful at all, and 5 corresponds to very useful. The average grade was found to be 4.61, and this result agrees with the conclusions discussed above.

6.4 Improvement of Understanding

This experiment aims to show that the use of automated explanation support improves the understanding of the domain for non-SCM experts. More importantly, it aims to show that this improvement is bigger than in the case where no automated explanation support is available. The experiment was designed to study the performance of non-SCM experts in two sets of similar tasks, and test their improvement for the second task. Two cases are compared: one where automated explanation support was provided for the first set of tasks and one where it was not.

6.4.1 Experimental Setup

This experiment was designed to test the performance of participants in two similar settings, and study any performance improvement involved. In order to guarantee the similarity of the two settings, similar questions were asked on similar types of problems in similar scenarios. Let us explain this in more detail: Two SC scenarios were used for this experiment, Scenario1 and Scenario3. As discussed in Section 6.2.1, the operation dynamics complexity of these scenarios is of the same scale. More specifically, they involve an SC simulation for the same time period and they include the same number of problematic situations of each type. Furthermore, there is a direct mapping between the propagation of individual problematic situations in the two scenarios. Three questions were asked for each scenario, covering the three question types mentioned in Section 6.2.2. Each question for Scenario1 was similar to a question for Scenario3. This was achieved by focusing on similar types of problems for the paired questions, which were similarly propagated in the two scenarios.

In this experiment, subjects were asked to answer scenario questions with or without automated explanation support. The process of answering a question in the two cases was as explained in Section 6.3.1. For each subject, two variables were measured for the answering of each scenario question: the time to answer the question and the correctness of the provided answer. The procedure and units of measurement were as described in Section 6.3.1.

	Group3	Group4
Scenario1	Y	N
Scenario3	N	N

Table 6.5: Availability of automated explanation support per group and scenario

The number of subjects in this experiment was 10, all of which were business experts without SCM expertise. These 10 subjects were split in two groups, Group3 and Group4, of equal sizes. The rationale behind distinguishing between two groups lies on the choice of a between-group comparison of performance improvement. All subjects answered all questions for the two scenarios, some with and some without automated explanation support. The availability of automated explanation support per group and scenario is visualised in Table 6.5, according to which members of Group3 were provided with automated explanation support for Scenario1, while members of Group4 were not. Moreover, neither of the two groups was provided with automated explanation support for Scenario3.

Before explaining how the improvement of performance was calculated for each subject, it is worth clarifying two matters. Firstly, some members of Group1 and Group2 for the experiment described in Section 6.3 also participated in the experiment described here, as members of Group3 and Group4, respectively. This brought practical advantages for the conduction of experiments, as a smaller number of participants needed to be recruited. Secondly, in this experiment we are interested in the relative rather than the absolute improvement of performance. This way we value higher the performance improvement of subjects that did not perform well in Scenario1 (e.g. a correctness improvement of 3 units is regarded as more important in the case where the initial performance was 2 than in the case where the initial performance was 6).

The improvement of performance (with respect to correctness of answers and time to provide an answer) for participants within Group4 was calculated as follows: Let β_i be the performance for each $Question_i$ of Scenario1, and δ_i the performance for each $Question_i$ of Scenario3 (where Scenario1-$Question_i$ and Scenario3-$Question_i$ are paired questions, as explained previously). The relative improvement for each $Question_i$ answered by each Group4 subject is then:

$\frac{\delta_i - \beta_i}{\beta_i}$, in the case of correctness-related performance

and $\frac{\beta_i - \delta_i}{361 - \beta_i}$, in the case of time-related performance

An average value of performance improvement was calculated over the three questions for each subject and each metric. This way, two metrics were available for each participant's performance relative improvement: his correctness improvement and his time-efficiency improvement.

Calculating the improvement of performance for members of Group3 was similar, except for one point: Group3 members answered the questions for Scenario1 with the use of the explanation system. Since the correctness and efficiency of explaining SC operation dynamics with the use of the explanation system is significantly higher than without it (as concluded from the experiment presented in Section 6.3), it would not be sensible nor relevant to compare the performance of Group3 participants for Scenario1 and Scenario3. Therefore, the improvement of performance for members of Group3 was calculated by comparing their performance for Scenario3 to the average performance of business experts when answering Scenario1 questions without the use of the explanation system. Note that this average was calculated not only over members of Group4, but over all business experts that were asked to answer Scenario1 questions without the use of the explanation system (either for this experiment or for the experiment presented in Section 6.3). Hence, calculating the performance improvement for members of Group3 was as follows: Let $\overline{\zeta_i}$ be the average performance for each Question$_i$ of Scenario1 over all business experts that did not use the explanation system, and γ_i the performance of Group3 participants for each Question$_i$ of Scenario3 (where Scenario1-Question$_i$ and Scenario3-Question$_i$ are paired questions). The relative improvement for each Question$_i$ answered by each Group3 subject is then:

$\frac{\gamma_i - \overline{\zeta_i}}{\overline{\zeta_i}}$, in the case of correctness-related performance

and $\frac{\overline{\zeta_i} - \gamma_i}{361 - \overline{\zeta_i}}$, in the case of time-related performance

An average value of performance improvement was calculated over the three questions for each subject and each metric, in the same way as for members of Group4. Hence, two metrics were available for each participant's performance relative improvement: his correctness improvement and his time-efficiency improvement.

6.4.2 Results

Tables 6.6 and 6.7 contain the average relative improvement of correctness and time-efficiency, respectively, for all experiment participants in two cases: where automated explanation support was initially available and where it wasn't. The columns of the tables illustrate the choice of a between-group comparison, while their rows reveal the sample sizes for the two cases compared. Based on these values, statistical tests were performed to evaluate the improvement of correctness and efficiency for explaining SC operation dynamics. These tests are discussed in the following two sections.

Correctness improvement for Group3	**Correctness improvement for Group4**
3.82	0.11
3.66	-0.17
3.94	0.11
2.95	0.33
0.32	1.44
Average = 2.94	Average = 0.367

Table 6.6: Relative improvement of correctness when automated explanation support was previously used (i.e. Group3) and when not (i.e. Group4)

Time-efficiency improvement for Group3	**Time-efficiency improvement for Group4**
0.67	7
7.59	35.01
4.76	0
3.08	46
1.80	2.67
Average = 3.579	Average = 18.136

Table 6.7: Relative improvement of time-efficiency when automated explanation support was previously used (i.e. Group3) and when not (i.e. Group4)

6.4.3 Test of Correctness Improvement

Statistical hypothesis testing using the t-distribution was conducted to evaluate the third research claim on improvement of correctness, and more specifically the claim that the improvement of correctness when automated explanation support was previously used is bigger than in the case where it was not. The hypothesis, null hypothesis, independent and dependent variables for this test are provided below.

- Hypothesis:
  The improvement of correctness of explanations of SC operation dynamics when the explanation system was previously used is bigger compared to the case where it was not previously used.
- Null Hypothesis:
  There is no difference in the improvement of correctness of explanations of SC operation dynamics when the explanation system was previously used and when it was not.
- Independent variables:
  – Data: 2 scenarios of complex and problematic SC operation (Scenario1, Scenario3)
  – Tasks:
  T1: answer questions on Scenario1 with the use of the explanation system and then answer questions on Scenario3 without the use of the explanation system
  T2: answer questions on Scenario1 and then on Scenario3 without the use of the explanation system
  – Subjects: 5 subjects for T1 and 5 subjects for T2, all of whom are business experts
- Dependent variables:
  – Correctness improvement when performing task T1 versus correctness improvement when performing task T2

A one-tailed t-test was performed to determine the t value and its corresponding p value in order to accept or reject the null hypothesis. A significance level of $p < 0.05$

was regarded as an acceptable condition to reject the null hypothesis. Given the choice of a between-group comparison of correctness improvement for T1 and T2, the two sample independent t-test was performed. The t value is given by the following equation:

$$t = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

where $\overline{x_i}$ is the mean of each sample, s_i is the standard deviation of each sample and n_i is the size of each sample.

The data of Table 6.6 was used for this test, and the calculated t-value was found to be 3.509. This value corresponds to a significance level of p=0.00856, which is much smaller than the significance level of 0.05. This means that the null hypothesis can be rejected. Hence we can conclude that the improvement of correctness of explanations on SC operation dynamics when the explanation system was previously used is significantly bigger compared to the case where it was not previously used.

6.4.4 Test of Efficiency Improvement

Given the data of Table 6.7, the average time-efficiency improvement of Group3 is smaller than the average time-efficiency improvement of Group4. For this reason, we decided to statistically test whether the improvement of time-efficiency when automated explanation support was previously used is smaller compared to the case where it was not previously used. The hypothesis, null hypothesis, independent and dependent variables for this test are provided below.

- Hypothesis:

  The improvement of time-efficiency for providing explanations of SC operation dynamics when the explanation system was previously used is smaller compared to the case where it was not previously used.

- Null Hypothesis:

  There is no difference in the improvement of time-efficiency for providing explanations of SC operation dynamics when the explanation system was previously used and when it was not.

- Independent variables:

  – Data: 2 scenarios of complex and problematic SC operation (Scenario1, Scenario3)

  – Tasks:

  T1: answer questions on Scenario1 with the use of the explanation system and then answer questions on Scenario3 without the use of the explanation system

  T2: answer questions on Scenario1 and then on Scenario3 without the use of the explanation system

  – Subjects: 5 subjects for T1 and 5 subjects for T2, all of whom are business experts

- Dependent variables:

  – Time-efficiency improvement when performing task T1 versus time-efficiency improvement when performing task T2

Similarly to the test presented in Section 6.4.3, a one-tailed two sample independent t-test was performed to determine the t value and its corresponding p value in order to accept or reject the null hypothesis. A significance level of $p < 0.05$ was regarded as an acceptable condition to reject the null hypothesis.

The data of Table 6.7 was used for this test, and the calculated t-value was found to be 1.541. This value corresponds to a significance level of $p=0.099$, which is higher than the significance level of 0.05. This means that the null hypothesis cannot be rejected. Therefore we cannot conclude that the improvement of efficiency when the explanation system was previously used is smaller compared to the case where it was not previously used.

We should emphasise that there was a positive improvement of efficiency when the explanation system was previously used. Since this was the case for all members of Group3, as shown in Table 6.7, we can conclude that the use of the explanation system improves the efficiency of non-SCM experts. This point is useful for the evaluation of the third research claim, and it is further discussed in the following section.

6.4.5 Discussion

An experiment with participants of business expertise was conducted to empirically evaluate the users' performance improvement achieved through the use of the explanation system. Experiment participants were split in two groups to answer similar questions on two similar SC operation scenarios. Answering questions on the second scenario was preceded by answering questions on the first one, with the use of the explanation system (for members of one group) or without (for members of the other group). The collected data was used for two statistical hypothesis tests over correctness and efficiency improvement. Based on the collected data and these tests we concluded the following:

- Explaining SC operation dynamics with the use of the explanation system improves the performance of non-SCM experts, with respect to their efficiency and correctness when providing relevant explanations.
- The improvement of correctness of explanations of SC operation dynamics when the explanation system was previously used is significantly higher compared to the case where it was not previously used.

Given these two points, one can conclude that the higher degree of correctness improvement achieved through the prior use of the explanation system does not come at the expense of time-efficiency. On the contrary, there is a parallel efficiency improvement. Hence, the second research claim is satisfied.

It is worth mentioning the magnitude of the difference of means between the relative improvement of performance in the two cases (i.e. with vs. without prior use of the explanation system). As it can be seen from Table 6.6, the average correctness improvement when the explanation system was previously used was more than eight times bigger compared to the case where the explanation system was not previously used.

The findings of this experiment have interesting implications with respect to the understanding of the domain for non-SCM experts. We have found that the prior use of the explanation system has a positive effect on the users' future performance for explaining SC operation dynamics without such support (i.e. based on simulation results). Simply put, non-SCM experts that have previously used the explanation

system are faster and more correct in their answers when analysing SC operation dynamics without the use of the explanation system. The fact that they are faster and more correct indicates that they have better understood the subject of SC operation dynamics. We, thus, believe that this suggests that the use of automated explanation support improves the understanding of the domain for non-SCM experts.

6.5 Analytical Evaluation of Maintainability and Reusability

In this section we aim to evaluate the third research claim, and thus show that the proposed knowledge-based approach for modelling, simulating and explaining SC operation allows for maintainability and reusability. Let us first clarify what is meant by these two terms. Maintainability in software engineering is defined as "the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment" (IEEE Std. 610.12, 1990). In this work we focus on adaptive maintenance, i.e. on the ease of modifying the functionality of a software system to adapt to a changed environment. Software reuse is the use of existing software artefacts to develop a new software system (Krueger, 1992). Hence, in the context of this thesis we understand software reusability as the ability of some artefact to be reused for some other application, different from the SC operation domain.

We analytically evaluate this claim with respect to three aspects of this work: (1) the SC operation input models, (2) the developed simulation system and (3) the developed explanation system. We first discuss some properties of these three aspects that contribute towards maintainability and reusability, such as modularity, cohesion, coupling, generality and declarative formalism. Then we explain how these properties support the maintainability and reusability of each aspect and we provide illustrating examples.

The SC operation models that can be specified following the modelling approach discussed in this thesis have the following properties:

- **Formal and declarative:** In Chapter 4 we presented the declarative specification of the modelling constructs through Prolog-based predicates.

The resulting model is formal; this means that it has clear execution semantics, as discussed in Chapter 5. The choice of a declarative approach also allows for a clear separation between the specified model and its execution semantics.

- **Generic:** The constructs for conceptualising and formalising SC operation are not pertaining to specific SC types or industries, and they can support a wide range of SC operation structures and behaviours (e.g. SC members' decision-making is captured by generic business rules instead of a fixed set of predefined policies). In their vast majority, they are general enough to be used for modelling additional domains, different from supply chains. For example, business processes can represent clinician activities.
- **Loosely-coupled:** The SC operation modelling constructs are clearly separated and independent from each other, both conceptually and with respect to their formalisation. There are two exceptions to this: the specification of inventory refers to specified entities, and business rules that serve as process preconditions are tightly coupled with the corresponding processes.

The simulation system that has been developed following the rule-based approach discussed in this thesis has the following properties:

- **Modular:** The components of the developed system, as presented in Section 5.1, are clearly separated. This means that each component makes sense when considered separately from the others (Robertson et al. 1991). For example, the workflow engine and the reasoning engine are clearly separated, both conceptually and functionally. It is worth mentioning that the modularity of the developed system is enabled through the modelling approach, which allows for models with clear structure.
- **Loosely-coupled:** The components of the developed system, as presented in Section 5.1, are independent from each other, meaning that there are weak interconnections between them (Yourdon and Constantine, 1979). The system's loose coupling is imposed by the modelling approach in the following way: The execution semantics of each construct, as specified in

Section 5.1, are independent from each other; since the internal implementation of each system component is based on the execution semantics of the corresponding construct, the functionality of each component is independent from each other. This means that a component may call the service offered by some other component without caring about its internal implementation. Let us demonstrate the loose coupling between the workflow engine and the reasoning engine through an appropriate example: Consider the case of a business process P that includes a business rule R within its preconditions. In order to execute P, the workflow engine needs to check the satisfaction of R, and thus calls the corresponding service offered by the reasoning engine (i.e. execute_precondBR, as presented in Section 5.1.2.2) without caring about how execute_precondBR is implemented.

- **Cohesive:** The developed simulation system is cohesive, as its components contain elements that are tightly related to one another (Yourdon and Constantine, 1979). This also means that the responsibilities of the system components are highly focused. For example, the purpose of the workflow engine is to execute business process models, thus bringing about the acting behaviour of SC members; all elements within the workflow engine fit conceptually and functionally within this purpose (e.g. the execution of actions such as creating entities is part of the SC members' acting behaviour). The modelling approach supports the high degree of cohesion in the following way: Each system component is implemented based on the execution semantics of the corresponding construct; hence all elements of the component are designed and built around aspects of the corresponding execution semantics, which are by definition tightly related.

- **Generic:** Most of the simulation system components are generic, in the sense that they are not specific to the SC operation domain, and thus could be used for simulating other domains. For example, the workflow engine could be used for the management of health operations. It is worth mentioning that the generic character of several system components is imposed by the formalisation of the corresponding modelling constructs, which is also of a generic nature.

The explanation system that has been developed following the logic-based approach discussed in this thesis has the following properties:

- **Generic:** The process of generating explanations (i.e. low-level explanations of SC operation and high-level explanations of problematic SC operation) is generic. This means that the functionality of the explanation system is not specific to the SC operation domain, and thus could also be used for different domains. There are two factors that contribute to the generic nature of the explanation system: Firstly, explanations are derived based on relevant information in the simulation log, which has the generic form fact(SimulationEvent, ListOfReasons, Timepoint). Secondly, the explanation derivation consists of generating proof trees given the generic fact/3 information; these proof trees are devised in a generic way, as already pointed out in Section 5.2.2.
- **Declarative:** The reasoning for generating explanations for different types of questions has been declaratively specified, as discussed in Section 5.2.2. The declarative logic programming language Prolog was used for the corresponding implementation, allowing for a direct and elegant implementation.

The above properties have been discussed, as they are known to contribute towards maintainability and reusability. Software maintainability is supported by modularity, loose coupling and high cohesion (Yourdon and Constantine, 1979). In addition, the existence of generic constructs, methods, and components within a software system may decrease the effort of evolving it to meet changing needs. Software reusability is facilitated through formal and generic models and procedures (Prieto- Díaz, 1993). Moreover, modularity, loose coupling and high cohesion ease the reuse of individual system components.

6.5.1 Input Model's Maintainability and Reusability

The SC operation input model is specified based on the modelling constructs identified in Chapter 4 and their execution semantics that support simulation and explanation, as discussed in Chapter 5. Note that by the term "input model" we refer

to an instantiated SC operation model, which can be used as an input for both simulation and explanation purposes.

Modifying the input model is needed when aspects of the studied SC operation are changed, and it involves removing or modifying existing elements of the input or adding new ones. The SC operation input model is *maintainable* in the sense that modifying it does not require much effort. This is understood along three points:

1. Modifying the input model does not affect much of the existing input specification.
2. Enriching the input model to include complex and specialised SC behaviours does not typically require additional system implementation.
3. The task of modifying the input model is conceptually straightforward.

The first point is supported by the modularity of the modelling constructs in two ways. Firstly, the loose coupling of constructs eases the process of removing or adding new elements in the simulation input. For example, adding a new policy for some SC member consists of simply adding the corresponding `br/4` predicate. Secondly, the explicit structure of the constructs and their clear, declarative linking allows modifying elements of the simulation input in a minimal way. For example, consider the case of SC partner change, where some SC member (e.g. `manufacturer`) decides to source from a different supplier (e.g. `supplier2` instead of `supplier1`). Modifying the simulation input accordingly involves simply updating the corresponding `data/3` information without modifying any additional sourcing behaviour elements: `data(manufacturer, product1_supplier, supplier1)` is updated to `data(manufacturer, product1_supplier, supplier2)` and this modification is "read" by the corresponding sourcing business processes, as the supplier information is not hard-coded into the businesses process specification.

The second point is supported by the generic nature of the modelling constructs and their formal and declarative representation. Consider the following example of complex flexibility decision-making for some SC member, which involves reacting to a situation where a combination of things may hold: either E or the combination of any case between A and B, and any case between C and D. This is expressed in propositional logic as follows: `((A or B) and (C or D)) or E`. This complex and

specialised SC behaviour can be captured through the specification of a flexibility business rule, the IFpart of which consists of the above logical expression.

The third point refers to the ease of correctly identifying the elements of the input model that need to be updated given some modification of the studied SC operation. This is an important issue, especially in the case where domain experts are in charge of specifying and modifying the simulation input. The choice of a formal modelling approach and the resulting explicit specification of constructs facilitate the identification of the corresponding elements of the input model. Furthermore, the declarative nature of the modelling approach separates the specification of constructs from their execution semantics; hence, when domain experts specify the simulation input, they do not need to worry about how the model is going to be run. Lastly, we believe that the names of formalised constructs are representative of the domain, and thus understandable by domain experts.

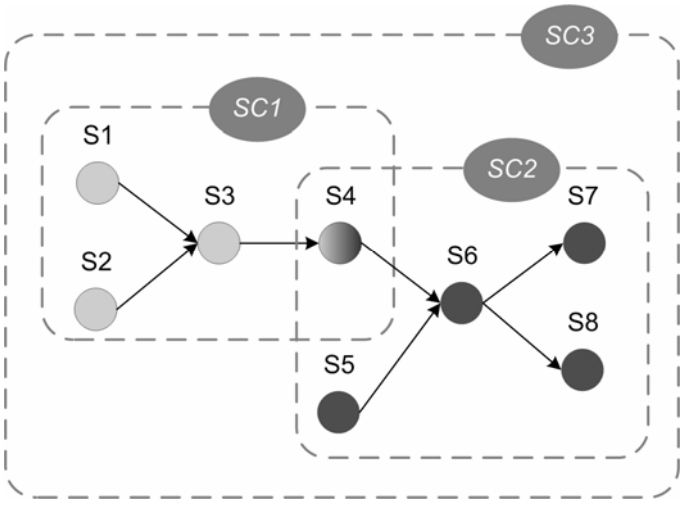

Figure 6.1: Merging supply chains SC1 and SC2 results into SC3. In order to specify the input model for SC3 we can reuse the input models for SC1 and SC2.

SC operation input model reusability refers to the degree to which specific parts of the model can be reused when specifying the input model for different supply chains or SC operation scenarios. The explicit structure of the constructs and their loose coupling support reusability, as they allow to easily identify, introduce and remove input model elements. We will now illustrate this point through an example. Consider supply chains SC1 and SC2 presented in Figure 6.1, and let's suppose that the simulation input for each of these has already been specified (i.e. Input1 and Input2 respectively). If we decide to merge these two supply chains into SC3 and

study the resulting behaviour as a whole, then the input model for SC3 fully reuses Input1 and Input2. In fact, no additional information needs to be specified for SC3 input model, i.e. $\texttt{Input3} = \texttt{Input1} \cup \texttt{Input2}$.

6.5.2 Simulation System's Maintainability and Reusability

The functionality of the simulation system stems from the modelling framework proposed in this thesis in two ways. Firstly, the simulation behaviours are based on the constructs formalised in Chapter 4 and their execution semantics defined in Chapter 5. Secondly, the system architecture was decided considering the structure of the formal model.

The functionality of the simulation system may be modified along two dimensions: On the one hand, modifications may be needed in order to simulate additional or modified constructs, and thus deal with a richer domain representation. On the other hand, new or modified simulation behaviours may be sought for the existing set of modelling constructs; this could include relaxing assumptions made when defining the model's execution semantics. The simulation system is *maintainable* in the sense that modifying it does not require much effort. This is understood along two points:

1. Maintaining the simulation system does not affect a big part of the existing implementation.
2. Extending the scope of the simulation system to deal with new or complementary aspects of the SCM domain does not require additional implementation, as long as these aspects follow the specified execution semantics.

The first point is supported by the system's modularity, as its loose coupling eases the process of introducing new components and removing or modifying existing ones. For example, modifying the simulation system to deal with funds (remember that funds were conceptualised and formalised in Chapter 4 but they were not implemented in the system) involves only one system component: the workflow engine would have to be modified to deal with funds-related process preconditions and actions. This minor modification would not affect any other system component.

Another example involves the introduction of a new component to perform business process analysis, and hence detect bottlenecks or unreachable points. The simulation system architecture would be modified as illustrated in Figure 6.2, and the new component introduction would not affect the internal implementation of any of the existing system components.

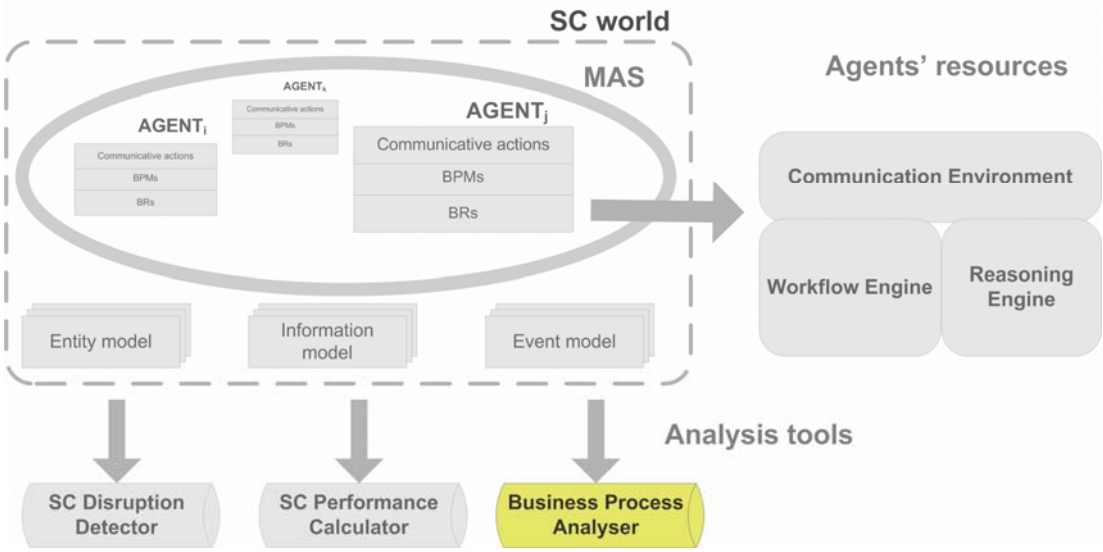

Figure 6.2: Introducing the new system component does not affect the internal implementation of other components

Two more properties of the simulation system support its maintainability with respect to the first point outlined above. Firstly, the high cohesion of individual components means that only a small number of components are modified, if not only one, when evolving the simulation system's functionality. Secondly, our declarative approach facilitates the direct modification of the execution semantics, as it allows for their explicit and separate definition from the specification of the modelling constructs. Let us illustrate these points through an example: Let's suppose that we want to relax the assumption of full trust between SC members when they communicate. This means that a received inform-message is added to an SC agent's knowledge base only if the sender of the message is someone trusted. Maintaining the simulation system to encompass the new requirement on trust involves modifying only one system component, i.e. the communication environment, and particularly the related definition of execution semantics. This modification is illustrated below, and it is worth making two remarks: Firstly, this modification does not affect the implementation of any other system component, mainly thanks to the high cohesion

of the communication component. Secondly, this modification does not require changing the definition of the involved constructs, thanks to the separate and declarative definition of the execution semantics.

```
read_message(MessageId, SenderId, Content):
IF (  received(MessageId)
      AND trustworthy(SenderId)
    )
THEN create_fact(Content)
```

The second point on maintainability is supported by the generic nature of the modelling constructs and the corresponding system components. For example, if the SCOR model is enriched to include additional processes, these are likely to be represented based on the formalisation discussed in Section 4.3.2.2. Consequently, and given the well-defined execution semantics for business processes, the workflow engine is generic enough to deal with the new SCOR-based processes, and thus no additional implementation is required. Another example involves extending the scope of the simulation environment to include decision-making on supplier selection. The criteria for selecting suppliers could be captured through business rules, and the corresponding reasoning could be simulated with the use of the reasoning engine.

Simulation system *reusability* refers to the degree to which specific components of the simulation system can be used in a different system, for a different application. The modularity of the simulation system, i.e. the high cohesion of its components and their loose coupling, support the system's reusability, as they facilitate the identification and extraction of simulation components in order to use them in a different setting. In addition, the generic nature of modelling constructs and system components broadens the range of candidate applications. We will now illustrate these aspects through an example. As already mentioned in Section 6.5.1, business processes can be used to represent clinician activities, and they can be simulated with the use of a workflow engine. Given its modularity, the workflow engine can thus be reused within a health informatics application. Another similar example is the use of business rules and the reasoning engine to simulate decision-making on Customer Relationship Management.

6.5.3 Explanation System's Maintainability and Reusability

The functionality of the explanation system is based on the execution semantics of the formalised constructs, as represented by causal information in the simulation log. In the case of high-level explanation of problematic SC operation, the generation of explanations is also based on the declarative causal model presented in Section 4.3.3.2.

The functionality of the explanation system may be modified along two dimensions: On one hand, modifications may be needed in order to explain additional or modified models (i.e. modified with respect to the modelling constructs and their execution semantics, as well as the causal model), and thus deal with a richer domain representation and simulation. On the other hand, new explanation behaviours may be sought for the existing formalised model; these are understood as the answering of additional types of questions. The explanation system is *maintainable* in the sense that modifying it does not require much effort. Let us break this claim in two points, referring to the two dimensions discussed above:

1. Maintaining the explanation system to deal with new or modified models does not require any additional implementation.
2. Maintaining the explanation system to answer new types of questions does not require much additional implementation.

The first point is supported by the declarative specification of execution semantics, as well as by the generic representation of causal information (i.e. through the generic predicate `fact/3` in the simulation log, and through the generic predicate `possible_reason/2` in the causal model). Let us illustrate this point through two examples. Firstly, let's suppose that the workflow engine has been modified to deal with funds, as discussed in the previous section. This modification includes the ability to add funds-related causal information in the simulation log, based on the business process execution semantics. Since this information is specified in the generic form of `fact/3`, no additional implementation is needed in order to explain funds-related behaviours. The second example involves explaining a modified model. Let's suppose that the simulation system has been modified to encompass the communication requirement on trust, as discussed in the previous section. Updating

the execution semantics for reading messages means that the related causal information is accordingly updated, i.e. any fact/3 within the simulation log that refers to reading a message includes the satisfied trust condition within its list of reasons. Nevertheless, the generic form of fact/3 is still preserved, and thus no additional implementation is needed in order to explain message-reading behaviours.

The second point on maintainability refers to answering new types of questions, by building on the ones already implemented. This is supported by the choice of the declarative logic programming language Prolog, which employs an intelligent unification strategy and allows for use of recursion with ease. For example, let's suppose that we want to identify common reasons for two situations. The notion of "common reason" can be expressed as follows:

$$reason(A, C) \wedge reason(B, C) \rightarrow common\_reason(A, B, C)$$

The implementation of common_reason(+A, +B, ?C) in Prolog is almost identical, as shown below. Building on the definition of reason/2, only three new lines of code are needed. It is also interesting to note the flexibility provided by this implementation: By giving the goal common_reason(a, b, C) we can obtain any common reason for two specific situations a and b.

```
common_reason(A, B, C):-
      reason(A, C),
      reason(B, C).
```

The explanation system is *reusable* in the sense that the components for generating explanations can be used in a different system, for a different application. For example, let's suppose that we want to explain behaviours for an emergency response scenario, simulated through a different simulation system, which keeps a simulation log with causal information of the form fact/3. Given this generic representation, we can use the current explanation system (without any modifications) to answer questions on the emergency response scenario. Hence, by utilising the generic reasoning process of the explanation system for answering different types of questions, we can identify effects and root causes of specific situations in an emergency response scenario.

6.5.4 Maintainability and Reusability Limitations

We recognise that the evaluation criteria of maintainability and reusability are not easy to measure and quantify. In Section 6.5 we have discussed these two properties, along with appropriate examples. By no means do we claim that SCOlog is fully maintainable and reusable; there are cases where maintaining or reusing the input model or the simulation and explanation system requires some engineering effort. For instance, extending SCOlog to address energy issues through measuring SCM carbon footprint would require additional implementation within the SC performance calculator. Nevertheless, we believe that the approach presented in this thesis satisfies these two criteria for typical modification and reuse cases in the context of SC operation.

6.5.5 Discussion

Analytical evaluation has been conducted on the third research claim on maintainability and reusability. We identified properties of SCOlog that contribute towards maintainability and reusability, and we provided illustrating examples along three dimensions of this work: (1) the specification of SC operation input models, (2) the developed simulation system and (3) the developed explanation system. Based on this analysis we concluded that the adopted knowledge-based approach for modelling, simulating and explaining SC operation scenarios allows for maintainability and reusability.

6.6 Satisfaction of Requirements & Limitations

6.6.1 Satisfaction of Requirements

In Section 2.1.5 we identified seven desired properties of a solution to the research problem. We now discuss how the approach proposed in this thesis, and consequently the implemented system, cover these properties.

1. **Holistic view:** The SC operation model described in Chapter 4 considers the entire SC network, thus representing the operation of several SC members. SC-wide behaviours can be simulated and overall SC performance can be

measured. The dynamics of system-wide SC operation can be explained, as illustrated in Chapter 5.

2. **Include SC disruptions:** The proposed modelling framework considers problematic SC operation; SC disruptions and low SC performance are explicitly represented. The simulation system architecture includes an SC disruption detector, and hence SC disruptions are identified at run time. Problematic SC operation can be explained at two levels of detail, thus identifying causal relationships between different SC disruptions, as well as interrelations of SC disruptions and SCM decisions and activities of different SC members. Hence, SC disruptions and SC operation dynamics are studied in an integrated way.

3. **Cover standard aspects of SC operation:** We adopt the SCOR model for modelling the behaviour of SC members. The SCOR model is a standard of the SCM domain (Bolstorff and Rosenbaum, 2012), and it covers the main facets of SC operation, such as product sourcing, making, delivering and returns. As far as SC disruptions are concerned, we consider problematic situations that are experienced from different sources: internally, from the supply side or the demand side. Furthermore, experiment participants of SCM expertise were asked to evaluate the severity of effects and the likelihood of occurrence of three addressed SC disruption types: delays, demand discrepancies and errors with items. A grade from 1 to 5 was assigned for each, where 1 is not severe/likely at all and 5 is very severe/likely, as shown in Appendix B. All three types of disruptions were evaluated as having severe effects; the average grade for each SC disruption type was, respectively: 4.71, 4.43 and 4.5. The likelihood of occurrence of the three types of disruptions was evaluated as medium to considerable; the average grade for each SC disruption type was, respectively: 4.21, 3.57 and 3.

4. **Deal with complex situations:** Complex SC structures, decision-making and acting behaviours can be modelled, simulated and explained. The modelling framework proposed in this thesis adopts an agent-oriented approach and

allows the specification of supply chains with a complex structure. Complex decision-making at different SC members can be captured through the use of if-then business rules. We have used FBPML to specify the acting behaviour of SC members; FBPML is an expressive language that can represent preconditions and effects of activities and that recognises different types of junctions. This way, complex SCM acting behaviours can be represented through FBPML-based business process models. These can also be simulated and explained, as illustrated through the example supply chain.

5. **Include flexibility aspects:** Recognising the trend of SC agility, we model flexibility decision-making as part of the thinking behaviour of SC members, and we declaratively specify it with the use of business rules (Section 4.3.2.1). Flexibility decisions and behaviours can, thus, be simulated and explained, as discussed in Chapter 5. The example supply chain used throughout this thesis includes such agility aspects, as an alternative supplier is used by Supplier4 in urgent situations.

6. **Facilitate what-if analysis:** An SC simulation system has been implemented as part of the research presented in this thesis. This simulation system can be used to experiment with different SC operation parameters and configurations, thus allowing for sensitivity analysis. What-if analysis is also supported by the maintainability of the specified simulation input models, as demonstrated in Section 6.5.1.

7. **Maintainability:** We have shown that the developed simulation and explanation system is characterised by maintainability (Sections 6.5.2 and 6.5.3). This means that modifying the simulation and explanation environment to incorporate SCM theoretical advances would not require much effort.

6.6.2 Limitations of Implemented Solution

We will now discuss some limitations of the implemented simulation and explanation environment. These limitations are not considered to be limitations of the research approach, but they are rather related to the actual implementation, as

presented in Chapter 5. The majority of these issues can be easily dealt with, and they were not addressed in the context of this PhD project due to time limitations.

Firstly, three aspects of the modelling framework presented in Chapter 4 were not fully implemented: funds, SC performance metrics and messages. Funds and fund-related process preconditions and actions were not included in the implementation. Their incorporation is, however, a straightforward task. As far as SC performance metrics are concerned, a subset of the metrics that were conceptualised in Section 4.2 are formalised and implemented as part of the SC performance calculator. The sending and receiving of inform-messages has been implemented, while other message types are not supported by the implementation. Nevertheless, inform-messages were found to be sufficient for dealing with typical SC operation scenarios.

Secondly, the implemented solution does not allow the control of two aspects of SC scenarios: priorities and non-fixed process cost. By priorities we refer to the priorities between business processes that are awaiting to be executed and that are competing for the same resource, priorities between competing business rules and priorities between resources that can be used by some business process execution. As far as process cost is concerned, the current implementation does not allow the simulation of business processes with variable cost (e.g. cost depending on the amount of entities dealt with by the process). Minor extension of the simulation and explanation environment would be sufficient to address these issues.

Thirdly, there are three BPM-related points that are not covered by the implemented workflow engine: validation, process decomposition and loops. Under the assumption of correct and conflict-free business process models, workflow validation is not required. Process decomposition is not implemented, and hence it is not possible to simulate processes at multiple levels at the same time. Loops within business process models cannot be dealt with by the implemented workflow engine. It is worth noting that we do not regard the last two points as considerable limitations, as one can simulate SCOR-based business process models without such issues; the example SC case illustrated this.

Finally, there is a low degree of usability of the implemented simulation and explanation environment. There is no graphical user interface available, and no

animation is provided at simulation run-time. These are standard features of commercial SC simulation tools, as mentioned in Chapter 3, and they are particularly useful when the end user is a supply chain manager, with no modelling expertise. Nevertheless, the system implementation presented in Chapter 5 is in the context of the PhD research project, and it is not intended for a direct use in industry. Increasing the usability of the implemented system would require considerable engineering time and effort, without directly contributing to the aim of this research. For this reason, we consider usability issues to be beyond the scope of this research.

6.7 Comparison to Related Work

In Chapter 3 we identified two research areas that implicitly address the problem of analysing SC operation dynamics: SC simulation and SC disruption analysis. In this section, we examine how the research presented in this thesis relates to these fields, comparing it to related work.

Recognising the usefulness of simulation techniques for studying complex systems, SC simulation is employed as part of the solution proposed in this thesis. The adoption of a declarative and rule-based approach for modelling and simulating SC operation facilitates the explanation of simulation results. This explanation facility is the most important strength of this work, compared to related work in SC simulation. As discussed in Section 3.1, existing commercial and research approaches in SC simulation treat simulation as a black box, and hence do not explain SC behaviours and SC performance results. Another advantage of this work is the clear modelling and analysis of problematic SC operation, as opposed to existing work in SC simulation, which does not explicitly address SC disruptions. In addition, this work allows the modelling, simulation and analysis of flexibility decision-making, thus capturing SC agility aspects; such issues are typically not incorporated in existing SC simulation models. We should make clear that we are not claiming that this work is generally superior to related work in SC simulation; existing SC simulation approaches have some undeniable advantages, such as high degree of usability, and the incorporation of variability and geographical aspects of supply chains. We believe, however, that the research presented in this thesis is

superior to related work in SC simulation with respect to the problem of analysing SC operation dynamics, given the above-mentioned strengths.

SCOlog allows for a high-level explanation of problematic SC operation, thus supporting SC disruption analysis. It also fills the three gaps of related work in the field, as identified in Section 3.2. Firstly, the proposed modelling approach is tailored to the SCM domain. A clear conceptualisation of SC operation is described, and modelling constructs are explicitly specified. To make the resulting models easier to build and understand, the SCOR model is used and typical inventory policies are captured. Secondly, this work allows the identification of root causes of low SC performance, as opposed to related work. This was illustrated through a running example in Section 5.2.2.3. Thirdly, the process of specifying SC operation input models is characterised by maintainability, as discussed in Section 6.5. This is a considerable advantage compared to existing approaches in SC disruption analysis, which often lead to large and difficult to manage SCM models.

The research presented in this thesis allows for a joint study of SC operation dynamics and SC disruptions. This means that SC disruptions are not analysed in isolation, as they are not considered to be rare, exceptional cases of SC operation. On the contrary, we recognise the increasing frequency of occurrence of disruptive events in modern supply chains, and we enable their analysis with respect to SC operational aspects.

6.8 Evaluation Summary

This chapter presented the evaluation framework and findings for the three research claims of this thesis. According to our first research claim, automated explanation support is useful for the task of explaining SC operation dynamics. In order to validate this claim, we conducted experiments with the participation of SCM and business experts. Based on these experiments, we found that the users' efficiency, correctness and certainty regarding explanations of SC operation dynamics that are provided when using the explanation system is significantly higher compared to the case of no automated explanation support.

According to our second research claim, the use of automated explanation support improves the explanation performance of non-SCM experts. We empirically evaluated this claim through experiments with the participation of business experts. Based on these experiments, we found that there is a positive improvement of users' correctness and efficiency for providing explanations when the explanation system is previously used. More importantly, the correctness improvement is significantly higher compared to the case of no prior explanation system use, without loss of efficiency. The demonstrated performance improvement of the users suggests that the use of automated explanation support improves the understanding of the domain for non-SCM experts.

According to our third research claim, a logic-based approach for modelling, simulating and explaining SC operation scenarios allows for maintainability and reusability. We validated this claim through analytical evaluation over three aspects: (1) the specification of SC operation input models, (2) the developed simulation system and (3) the developed explanation system. Illustrative examples were provided for each of these dimensions.

Furthermore, we discussed how SCOlog satisfies certain SCM domain requirements, as identified in Chapter 2. A direct comparison to related work was also provided, thus highlighting the strengths of this research.

Chapter 7

Conclusions and Future Work

The research presented in this thesis tackled the problem of analysing supply chain operation dynamics, a problem that has been understudied by literature so far. Yet, understanding SC-wide operation dynamics is highly important for coordinating SC activities, and ultimately integrating supply chains. To our knowledge, SCOlog is the first attempt for an explicit and thorough solution to this problem.

SC simulation is an area that captures, to some extent, SC operation dynamics. However, existing work in SC simulation provides only a partial solution to the problem of analysing SC operation dynamics, as it does not allow for the direct explanation of simulation results. In order to fill this gap, SCOlog employs a knowledge-based approach to SC modelling and simulation. This way automated explanation support is provided, which is found to give a useful insight into SC operation dynamics. Recognising the trend of SC agility, SCOlog incorporates flexibility decision-making and acting. SC flexibility aspects are addressed by a limited number of SC simulation efforts, and they are widely neglected in SC coordination studies (Chan and Chan, 2010).

In this work we consider SC disruptions to be an integral part of SC operation dynamics. SCOlog explicitly addresses SC disruptions, making it possible to simulate the occurrence of disruptive events during SC operation, as well as to analyse their causes and effects. There are some recent research efforts towards the analysis of SC disruptions, such as (Liu et al. 2007). However, these approaches do not allow for the identification of root causes of low SC performance and they suffer

from usability problems, as the proposed modelling approaches are not tailored to the SCM domain and they are not maintainable. SCOlog fills these gaps, and provides an alternative solution to those presented in the literature, while jointly considering aspects of SC operational behaviours and SC disruptions.

7.1 Thesis Summary

This thesis presented a logic-based approach for analysing SC operation dynamics, named SCOlog. An overview of the area of Supply Chain Management was provided in Chapter 2, highlighting the importance of understanding system-wide SC operation dynamics. This is a complex problem that requires an intelligent solution. To this end, SCOlog employs Artificial Intelligence techniques, such as knowledge-based reasoning, workflows and intelligent agents. Background information on these areas was provided in Chapter 2.

Chapter 3 presented related work on SC operation dynamics classified into two research streams: SC simulation and SC disruption analysis. We discussed the appropriateness of simulation techniques for capturing the dynamics of complex systems, and presented existing commercial and research approaches. The main gap identified is the lack of an explanation facility for SC simulation results; the problem of analysing SC operation dynamics is, thus, not explicitly addressed by SC simulation literature. On the other hand, research in SC disruption analysis explicitly addresses the interrelations between disruptive events that occur in a supply chain network. The main gap identified in this stream of research involves usability aspects, as existing approaches are not tailored to the SCM domain.

In order to fill this gap, we conceptualised SC operation in Chapter 4 by taking structural and behavioural aspects into account, as well as problematic SC operation. We formalised the domain by perceiving SC members as logic-based intelligent agents consisting of three layers: (1) reasoning layer, represented through business rules, (2) process layer, represented through business processes and (3) communication layer, represented through communicative actions. Structural and disruption-related constructs were declaratively formalised, and a logic-based causal model was defined, capturing possible reasons for the occurrence of problematic

situations. In order to illustrate aspects of the modelling framework, we discussed the model of an example supply chain.

The approach of SCOlog to addressing the identified gap in SC simulation literature was detailed in Chapter 5, consisting of a simulation and explanation framework. As far as simulation is concerned, we designed a simulation system and specified the execution semantics of the formal model presented in Chapter 4. A rule-based representation of these semantics was adopted in order to enable the explanation of simulated behaviours. A simulation algorithm was provided and aspects of the implemented SC simulation system were discussed, along with illustrating examples of its use. As far as explanation is concerned, we provided a mechanism for translating the specified execution semantics into grounded, low-level causal information, which was added to the simulation log during run-time. The generation of explanations of SC operation was, thus, driven by this information. We also presented a framework for explaining problematic SC operation at a higher level of detail, based on the causal model defined in Chapter 4. The implementation of an appropriate explanation system was discussed, and its use was demonstrated for the example supply chain.

Chapter 6 provided a thorough empirical and analytical evaluation of our work with respect to the three research claims. Appropriate experiments were designed and conducted for validating the first two research claims, in which participants of SCM and business expertise were involved. The experiment for the first claim investigated the performance of subjects when explaining SC operation dynamics. This performance was measured with respect to the participants' efficiency, correctness and certainty, and it was found to be significantly higher when using the explanation system compared to the case of no explanation system use. This way, the usefulness of automated explanation support for SCM experts was demonstrated. The experiment for the second research claim demonstrated the usefulness of automated explanation support for business experts with no SCM expertise. Based on this experiment we found that the performance of subjects when explaining SC operation dynamics was improved with the use of automated explanation support, as indicated by their correctness and efficiency. More importantly, the correctness improvement was significantly higher compared to the case of no prior explanation system use,

without loss of efficiency. These findings suggest that the use of automated explanation support improves the understanding of the domain for non-SCM experts. The third research claim on maintainability and reusability was analytically evaluated. We first identified some properties of the specified SC operation input models, the developed simulation system and the developed explanation system that contribute towards maintainability and reusability, such as modularity, declarative and generic nature. We then discussed aspects of maintainability and reusability, which were demonstrated through appropriate examples. In this chapter, we also compared and contrasted our work to competing approaches, thus demonstrating its strengths.

7.2 Contributions

This thesis has made the following scientific contributions:

- **Formal model of SC operation:** SC operation has been formally modelled taking conceptual and representational issues into account. We identified structural, behavioural and disruption-related constructs in order to conceptualise SC operation in a way that is tailored to the domain. We provided a declarative specification of the conceptual model, bringing benefits of maintainability and reusability. The resulting model is comprehensive and can sufficiently describe real domains. A causal model of problematic SC operation has also been defined, capturing causal relationships between different types of SC disruptions and low SC performance.
- **Executable model of SC operation and implemented simulation system:** We provided a rule-based specification of the execution semantics of the formal model, along with an appropriate simulation algorithm. A maintainable and reusable simulation environment has been implemented for analysing and experimenting with different SC operation scenarios.
- **Mechanism for generating explanations of SC operation and implemented explanation system:** A framework for explaining dynamic SC

operational behaviours has been provided based on the transparent specification of execution semantics. Utilising the defined causal model of problematic SC operation, we devised a method for diagnosing and explaining problematic SC operation. We have also implemented a maintainable and reusable explanation system for the automatic generation of explanations of normal and problematic SC operation. This automated explanation support has been found to be useful to both SCM experts and non-experts.

Given the multidisciplinary nature of this work, we make contributions to different scientific fields:

- **Supply Chain Management:** This is the first attempt to explicitly and thoroughly analyse SC operation dynamics. We contribute to the area of SC modelling through the formal definition of an SC operation model. We also contribute to the area of SC disruption analysis through a maintainable and context-aware method for analysing problematic SC operation. A practical contribution to the field is made through the implemented simulation and explanation systems.
- **Simulation:** We provide a new approach to simulating distributed and decentralised systems that consist of members with independent reasoning, acting and social capabilities. We also provide a generic mechanism for the automatic derivation of explanations about simulated behaviours. The implemented simulation and explanation systems can serve as solutions against which different simulation approaches can be evaluated.
- **Knowledge-Based Systems:** We describe a generic mechanism for explaining the operation of complex systems in an automated and maintainable way. The implemented explanation system is considered to be useful by users and it is a practical contribution to the field.

7.3 Limitations

We identify three main limitations of the research presented in this thesis. Firstly, stochastic aspects of SC operation are not considered. The assumption of deterministic SC operation is a strong assumption. Stochastic parameters of SC operation could involve final demand, transportation times, the availability of production resources, etc. In order to deal with the high degree of complexity of the problem, we decided not to include stochastic aspects in this work. Given that no other explicit solution to the studied problem was available, we found it more important to investigate the feasibility of a solution and its usefulness in a practical setting, even if that would be under the assumption of determinism. Having answered these questions, one can move on to incorporate variability aspects in our work. We identify two main directions towards such an extension: The first direction involves simulating and explaining SC operation models that include probabilities. This would require only minor modification of SCOlog. The second direction involves simulating multiple instances of SC scenarios, so as to get representative results (i.e. given stochasticity). The challenge here is providing aggregated explanation over multiple simulated scenarios. To this end, major modification of SCOlog's explanation mechanism would be needed.

Secondly, this work has focused on analysing the operation dynamics of manufacturing supply chains, i.e. supply chains developed around tangible products. SCOlog can also be used in the context of service supply chains, but certain characteristics that are particular to service supply chains are not currently addressed. We believe that this is not a strong limitation for two reasons: (a) The vast majority of SCM studies focus on manufacturing supply chains (Burgess et al. 2006; Sengupta et al. 2006), and (b) future work towards analysing the operation dynamics of service supply chains could be extensively based on the approach presented in this thesis.

Thirdly, SCOlog provides an approach to the analysis of operation dynamics of generic supply chains. This was a deliberate research design choice, as explained in Section 4.1. The advantage of this design decision is the generality of the solution and the corresponding wide audience. The price to be paid is that SCOlog may not satisfy some specific requirements of particular business sectors (e.g. quality issues,

which are important in food SCs, are not incorporated in the model). Extending SCOlog to address such issues would be an interesting topic to explore.

7.4 Future Work

We identify the following avenues of future research in order to advance the provided solution to the studied research problem:

- **Stochastic aspects:** As mentioned in the previous section, this work makes the assumption of deterministic SC operation. One could relax this assumption and include stochastic aspects in SCOlog. To this end, probabilistic modelling approaches (Bishop, 2006) can be useful. As discussed in the previous section, simulating and explaining SC operation models that include probabilities can be achieved with minor extension of our work. Further work would be needed to extend SCOlog's explanation mechanism in order to provide aggregated explanation over multiple simulated scenarios.
- **Industry-specific and service supply chains:** There is opportunity to extend this work to address aspects that are specific to particular industries or types of supply chains. Given the considerable interest in food supply chain management (Bourlakis and Weightman, 2004), SCOlog can be extended to address specific requirements on food SC modelling, such as the ones discussed by van der Vorst et al. (2009). There is also growing interest in service supply chains, given the reported increasing importance of the service sector (Ellram et al. 2004) and the high degree of servitisation of manufacturing (Baines et al. 2009). Extending SCOlog to address structural differences in service supply chains, such as heterogeneity and simultaneity (Baltacioglu et al. 2007), would therefore be timely. A candidate focus area could be telecommunication supply chains, which have been found to have complex dynamics and suffer from amplification problems (Akkermans and Vos, 2003).

- **Green aspects:** Within the wider global push towards a green economy, green supply chain management is an emerging theme in both theory and practice (Srivastava, 2007). Advancing SCOlog to incorporate sustainability aspects would, thus, be both timely and beneficial. A first step towards this direction should address green operations, such as waste management and reverse logistics, as well as green metrics, such as carbon footprint.
- **Implementation and large-scale supply chains:** The full potential of the approach proposed in this thesis can be achieved by addressing the implementation limitations discussed in Section 6.6.2. This includes further implementation for funds, additional message types and SC performance metrics, as well as dealing with priorities and some workflow-related engineering. At the same time, SCOlog can be employed for studying the operation dynamics of large-scale, complex supply chains. In this context, it is possible to perform multi-level analysis through a hierarchical view of supply chains and their operation. Conducting a case study in such a setting can prove beneficial not only for the involved organisations, but also for advancing and adjusting the current solution to real-world, large-scale problems.

It would be interesting to investigate the applicability of SCOlog to new problems and domains. Further directions of future work in this context are the following:

- **Teaching SCM:** We have empirically shown that SCOlog improves the explanation performance of non-SCM experts, thus suggesting an improvement of their understanding of the domain. This is a quality that is promising within an educational context. Most of the – relatively few – SCM teaching tools are based on the beer game (Sterman, 1989) and do not cover aspects of SC operation dynamics nor issues related to the propagation of SC disruptions. We believe that there is great potential for the developed simulation and explanation systems to be jointly used as a teaching tool for SCM. We understand that this would require considerable engineering effort for improving the usability of the existing systems. In order to provide a comprehensive solution to SCM education, one could integrate SCOlog with a

capability for teaching basic principles and introductory topics in supply chain management. To this end, we recognise ontologies (Gómez-Pérez et al. 2004) as a useful technology for supporting the teaching of fundamental SCM concepts.

- **SC planning and configuration:** The provided mechanism for generating explanations of simulated behaviours can be adjusted to the context of other SCM problems, such as system-wide planning and configuration. Analysing SC operation dynamics for dynamically reconfigurable supply chains could also be valuable. Agent-based techniques have been successfully employed in the past for SC formation problems (Piramuthu, 2005), and hence logic-based intelligent agents with improved reasoning capabilities could be useful in such a setting.
- **Beyond SCM – Health informatics:** Investigating the applicability of SCOlog to problems within the medical domain could be highly beneficial. Hospitals and clinical environments are complex systems, in which many actors with different levels of responsibility think, act and interact. The three-layered agent-oriented modelling approach of SCOlog, could thus, be relevant. Furthermore, it would be interesting to test whether the framework for high-level explanation of SC disruptions can be employed for the analysis of disruptive events in a medical environment. Lastly, collaboration aspects could also be considered; the work by Grando et al. (2011) could be of use towards this direction.

7.5 Concluding Remarks

This thesis presented a logic-based solution to the problem of analysing supply chain operation dynamics. A modelling framework was described for capturing and representing fundamental aspects of system-wide SC operation, allowing for the definition of declarative, maintainable models that are tailored to the SCM domain. A rule-based approach to simulation was employed, based on which an appropriate simulation environment was developed. Given this approach, we designed and implemented a mechanism for generating explanations of SC operation at two levels

of detail. This automated explanation support was empirically evaluated, and it was found to provide a useful insight into SC operation dynamics for both SCM experts and non-SCM business experts. The adoption of a logic-based approach brought advantages of maintainability and reusability, which are of considerable value in a rapidly-changing business environment. We believe that this work can serve as a basis for exploring further SCM problems and for studying alternative domains with complex dynamics.

Bibliography

Akanle, O. and Zhang, D. (2008). Agent-based model for optimising supply-chain configurations. *International Journal of Production Economics*, 115(2):444–460.

Akkermans, H. (2001). Emergent supply networks: System dynamics simulation of adaptive supply agents. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, Washington, DC, USA. IEEE Computer Society.

Akkermans, H. and Vos, B. (2003). Amplification in service supply chains: an exploratory case study from the telecom industry. *Production and Operations Management*, 12(2):204–223.

Allwood, J. M. and Lee, J.-H. (2005). The design of an agent for modelling supply chain network dynamics. *International Journal of Production Research*, 43(22):4875–4898.

Anderson, D., Sweeney, D., Williams, T., Freeman, J., and Shoesmith, E. (2009). *Statistics for Business and Economics*. Cengage Learning, London, UK, International edition.

Angerhofer, B. and Angelides, M. (2000). System dynamics modelling in supply chain management: research review. In Joines, J., Barton, R., Kang, K., and Fishwick, P., editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 342–351.

Archibald, G., Karabakal, N., and Karlsson, P. (1999). Supply chain vs. supply chain: using simulation to compete beyond the four walls. In Farrington, P., Nembhard, H., Sturrock, D., and Evans, G., editors, *Proceedings of the 1999 Winter Simulation Conference*, pages 1207–1214.

Arns, M., Fischer, M., Kemper, P., and Tepper, C. (2002). Supply chain modelling and its analytical evaluation. *Journal of the Operational Research Society*, 53(8):885–894.

Axsäter, S. (2006). *Inventory control*. Springer Science, New York, USA, 2nd edition.

Aydemir, A., de Vree, J., Brekelmans, W., Geers, M., Sillekens, W., and Werkhoven, R. (2005). An adaptive simulation approach designed for tube hydroforming processes. *Journal of Materials Processing Technology*, 159(3):303–310.

Bagchi, S., Buckley, S. J., Ettl, M., and Lin, G. Y. (1998). Experience using the IBM supply chain simulator. In Medeiros, D., Watson, E., Carson, J., and Manivannan, M., editors, *Proceedings of the 1998 Winter Simulation Conference*, pages 1387–1394.

Baines, T., Lightfoot, H., Benedettini, O., and Kay, J. (2009). The servitization of manufacturing: A review of literature and reflection on future challenges. *Journal of Manufacturing Technology Management*, 20(5):547–567.

Bajec, M. and Krisper, M. (2005). A methodology and tool support for managing business rules in organisations. *Information Systems*, 30(6):423–443.

Baltacioglu, T., Ada, E., Kaplan, M. D., Yurt, O., and Kaplan, C. Y. (2007). A new framework for service supply chains. *The Service Industries Journal*, 27(2):105-124.

Barnett, M. W. and Miller, C. J. (2000). Analysis of the virtual enterprise using distributed supply chain modeling and simulation: an application of e-SCOR. In Joines, J., Barton, R., Kang, K., and Fishwick, P., editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 352–355.

Barratt, M. (2004). Understanding the meaning of collaboration in the supply chain. *Supply Chain Management: An International Journal*, 9(1):30–42.

Basu, A. and Kumar, A. (2002). Research commentary: Workflow management issues in e-business. *Information Systems Research*, 13(1):1–14.

Beamon, B. (1999). Measuring supply chain performance. *International Journal of Operations & Production Management*, 19(3):275–292.

Bearzotti, L. A., Salomone, E., and Chiotti, O. J. (2012). An autonomous multi-agent approach to supply chain event management. *International Journal of Production Economics*, 135(1):468–478.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blackhurst, J., Craighead, C. W., Elkins, D., and Handfield, R. B. (2005). An empirically derived agenda of critical research issues for managing supply-chain disruptions. *International Journal of Production Research*, 43(19):4067–4081.

Bodendorf, F. and Zimmermann, R. (2003). Proactive supply-chain event management with agent technology. *International Journal of Electronic Commerce*, 9(4):58–89.

Bolstorff, P. and Rosenbaum, R. (2012). *Supply chain excellence: A handbook for dramatic improvement using the SCOR model*. AMACOM, New York, USA, 3rd edition.

Bourlakis, M. and Weightman, P. (2007). *Food Supply Chain Management*. Blackwell Publishing Ltd.

Browne, P. (2009). *JBoss Drools Business Rules*. Packt Publishing.

Burgess, K., Singh, P., and Koroglu, R. (2006). Supply chain management: A structured literature review and implications for future research. *International Journal of Operations & Production Management*, 26(7):703–729.

Butner, K. (2010). The smarter supply chain of the future. *Strategy & Leadership*, 38(1):22–31.

Chan, H. K. and Chan, F. T. (2010). A review of coordination studies in the context of supply chain dynamics. *International Journal of Production Research*, 48(10):2793–2819.

Chandra, C. and Grabis, J. (2007). *Supply Chain Configuration: Concepts, Solutions and Applications*. Springer Verlag.

Chatfield, D. C., Harrison, T. P., and Hayya, J. C. (2006). SISCO: An object-oriented supply chain simulation system. *Decision Support Systems*, 42(1):422–434.

Chen-Burger, Y.-H., Tate, A., and Robertson, D. (2002). Enterprise modelling: A declarative approach for FBPML. In *European Conference of Artificial Intelligence (ECAI), Knowledge Management and Organisational Memories Workshop*.

Choi, T., Dooley, K., and Rungtusanatham, M. (2001). Supply networks and complex adaptive systems: Control versus emergence. *Journal of Operations Management*, 19(3):351–366.

Chopra, S. and Meindl, P. (2003). *Supply Chain Management. Strategy, Planning & Operation*. Prentice Hall.

Chow, H. K., Choy, K. L., Lee, W., and Lau, K. (2006). Design of a RFID case-based resource management system for warehouse operations. *Expert Systems with Applications*, 30(4):561–576.

Christopher, M. (2000). The agile supply chain: Competing in volatile markets. *Industrial Marketing Management*, 29(1):37–44.

Christopher, M. and Lee, H. (2004). Mitigating supply chain risk through improved confidence. *International Journal of Physical Distribution & Logistics Management*, 34(5):388–396.

Chwif, L., Barretto, M. R. P., and Saliby, E. (2002). Supply chain analysis: spreadsheet or simulation? In Yucesan, E., Chen, C.-H., Snowdon, J., and Charnes, J., editors, *Proceedings of the 2002 Winter Simulation Conference*, pages 59–66.

Clocksin, W. and Mellish, C. (2003). *Programming in Prolog*. Springer, New York, USA, 5th edition.

Collins, J., Ketter, W., and Sadeh, N. (2010). Pushing the limits of rational agents: The trading agent competition for supply chain management. *AI Magazine*, 31(2):63–80.

Cope, D. (2008). *Automatic generation of supply chain simulation models from SCOR based ontologies*. PhD thesis, Department of Industrial Engineering and Management Systems, University of Central Florida.

Cope, D., Fayez, M., Mollaghasemi, M., and Kaylani, A. (2007). Supply chain simulation modeling made easy: An innovative approach. In Henderson, S., Biller, B., Hsieh, M., Shortle, J., Tew, J., and Barton, R., editors, *Proceedings of the 2007 Winter Simulation Conference*, pages 1887–1896.

Corbett, C. J. (2001). Stochastic inventory systems in a supply chain with asymmetric information: Cycle stocks, safety stocks, and consignment stock. *Operations Research*, 49(4):487–500.

Craighead, C. W., Blackhurst, J., Rungtusanatham, M. J., and Handfield, R. B. (2007). The severity of supply chain disruptions: Design characteristics and mitigation capabilities. *Decision Sciences*, 38(1):131–156.

Davenport, T. (1993). *Process innovation: reengineering work through information technology*. Harvard Business Press, Boston, MA, USA.

Dong, J., Ding, H., Ren, C., and Wang, W. (2006). IBM SmartSCOR – A SCOR based supply chain transformation platform through simulation and optimization techniques. In Perrone, L.,Wieland, F., Liu, J., Lawson, B., Nicol, D., and Fujimoto, R., editors, *Proceedings of the 2006 Winter Simulation Conference*, pages 650–659.

Doukidis, G. I. and Angelides, M. C. (1994). A framework for integrating artificial intelligence and simulation. *Artificial Intelligence Review*, 8(1):55–85.

Drzymalski, J. and Odrey, N. (2008). Supervisory control of a multi-echelon supply chain: A modular petri net approach for inter-organizational control. *Robotics and Computer-Integrated Manufacturing*, 24(6):728–734.

Ellram, L. M., Tate, W. L., and Billington, C. (2004). Understanding and managing the services supply chain. *Journal of Supply Chain Management*, 40(4):17–32.

Fox, M., Barbuceanu, M., and Teigen, R. (2000). Agent-oriented supply-chain management. *International Journal of Flexible Manufacturing Systems*, 12(2):165–188.

Friedman-Hill, E. (2003). *Jess in Action: Java Rule-Based Systems*. Manning Publications Co., Greenwich, CT, USA.

Georgakopoulos, D., Hornick, M., and Sheth, A. (1995). An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153.

Giarratano, J. C. and Riley, G. (1998). *Expert Systems*. PWS Publishing Co., Boston, MA, USA, 3rd edition.

Giunipero, L., Hooker, R., Joseph-Matthews, S., Yoon, T., and Brudvig, S. (2008). A decade of SCM literature: past, present and future implications. *Journal of Supply Chain Management*, 44(4):66–86.

Gómez-Pérez, A., Fernández-López, M., and Corcho, O. (2004). *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Verlag.

Goutsos, S. and Karacapilidis, N. (2004). Enhanced supply chain management for ebusiness transactions. *International Journal of Production Economics*, 89(2):141–152.

Graml, T., Bracht, R., and Spies, M. (2008). Patterns of business rules to enable agile business processes. *Enterprise Information Systems*, 2(4):385–402.

Grando, M. A., Peleg, M., Cuggia, M., and Glasspool, D. (2011). Patterns for collaborative work in health care teams. *Artificial Intelligence in Medicine*, 53(3):139–160.

Grüninger, M. and Menzel, C. (2003). The process specification language (PSL) theory and applications. *AI Magazine*, 24(3):63–74.

Gunasekaran, A., Patel, C., and Tirtiroglu, E. (2001). Performance measures and metrics in a supply chain environment. *International Journal of Operations & Production Management*, 21(1/2):71–87.

Harrison, A. and van Hoek, R. (2008). *Logistics Management and Strategy: Competing through the supply chain*. Prentice Hall, Harlow, Essex, UK, 3rd edition.

Harrison, J., Lin, Z., Carroll, G., and Carley, K. (2007). Simulation modeling in organizational and management research. *The Academy of Management Review*, 32(4):1229–1245.

Hines, P., Holweg, M., and Rich, N. (2004). Learning to evolve: a review of contemporary lean thinking. *International Journal of Operations & Production Management*, 24(10):994–1011.

Hollingsworth, D. (1994). The workflow reference model. Technical Report TC00-1003, Workflow Management Coalition.

Holweg, M. and Bicheno, J. (2002). Supply chain simulation – a tool for education, enhancement and endeavour. *International Journal of Production Economics*, 78(2):163–175.

Holweg, M. and Pil, F. K. (2008). Theoretical perspectives on the coordination of supply chains. *Journal of Operations Management*, 26(3):389–406.

Hwarng, H. B. and Xie, N. (2008). Understanding supply chain dynamics: A chaos perspective. *European Journal of Operational Research*, 184(3):1163–1178.

IEEE Std., 610.12, (1990). IEEE standard glossary of software engineering terminology. IEEE Computer Society Press, Los Alamitos, CA.

ILOG (2005). ILOG JRules: Leading the way in business rule management systems. White Paper.

Intelligent Systems Laboratory, Swedish Institute of Computer Science (2003). SICStus Prolog User's Manual, Release 3.10.1.

Isermann, R. (2006). *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*. Springer Verlag.

Isiklar, G., Alptekin, E., and Bykzkan, G. (2007). Application of a hybrid intelligent decision support model in logistics outsourcing. *Computers & Operations Research*, 34(12):3701–3714.

Ivanov, D., Sokolov, B., and Kaeschel, J. (2010). A multi-structural framework for adaptive supply chain planning and operations control with structure dynamics considerations. *European Journal of Operational Research*, 200(2):409–420.

Kalpic, B. and Bernus, P. (2002). Business process modelling in industry – the powerful tool in enterprise management. *Computers in Industry*, 47(3):299–318.

Kim, J. and Rogers, K. (2005). An object-oriented approach for building a flexible supply chain model. *International Journal of Physical Distribution & Logistics Management*, 35(7):481–502.

Kleijnen, J. (2005). Supply chain simulation tools and techniques: a survey. *International Journal of Simulation and Process Modelling*, 1(1-2):82–89.

Krueger, C. W. (1992). Software reuse. *ACM Computing Surveys*, 24(2):131–183.

Lambert, D. (2008). *Supply Chain Management: Processes, Partnerships, Performance*. Supply Chain Management Institute.

Lambert, D. M. and Cooper, M. C. (2000). Issues in supply chain management. *Industrial Marketing Management*, 29(1):65–83.

Lambert, D. M., Garca-Dastugue, S. J., and Croxton, K. L. (2005). An evaluation of process-oriented supply chain management frameworks. *Journal of Business Logistics*, 26(1):25–51.

Lawrynowicz, A. (2007). Production planning and control with outsourcing using artificial intelligence. *International Journal of Services and Operations Management*, 3(2):193–209.

Lee, H., Padmanabhan, V., and Whang, S. (1997). Information distortion in a supply chain: The bullwhip effect. *Management Science*, 43(4):546–558.

Lee, H. L. and Whang, S. (2004). E-business and supply chain integration. In Harrison, T. P., Lee, H. L., and Neale, J. J., editors, *The Practice of Supply Chain Management: Where Theory and Application Converge*, volume 62 of International Series in Operations Research & Management Science, pages 123–138. Springer US.

Li, Z., Kumar, A., and Lim, Y. (2002). Supply chain modelling – a co-ordination approach. *Integrated Manufacturing Systems*, 13(8):551–561.

Liang,W.-Y. and Huang, C.-C. (2006). Agent-based demand forecast in multi-echelon supply chain. *Decision Support Systems*, 42(1):390–407.

Liu, J., Wang, W., Chai, Y., and Liu, Y. (2004). Easy-SC: a supply chain simulation tool. In Ingalls, R., Rossetti, M., Smith, J., and Peters, B., editors, *Proceedings of the 2004 Winter Simulation Conference*, pages 1373–1378.

Liu, J., Zhang, S., and Hu, J. (2005). A case study of an inter-enterprise workflow-supported supply chain management system. *Information & Management*, 42(3):441–454.

Liu, R., Kumar, A., and van der Aalst, W. (2007). A formal modeling approach for supply chain event management. *Decision Support Systems*, 43(3):761–778.

LlamaSoft Inc. (2012). Supply Chain Guru.

Longo, F. and Mirabelli, G. (2008). An advanced supply chain management tool based on modeling and simulation. *Computers & Industrial Engineering*, 54(3):570–588.

Lummus, R. and Vokurka, R. (1999). Defining supply chain management: A historical perspective and practical guidelines. *Industrial Management & Data Systems*, 99(1):11–17.

Manataki, A. (2007). *A knowledge-based analysis and modelling of Dell's supply chain strategies*. Master's thesis, School of Informatics, University of Edinburgh.

Mayer, R., Menzel, C., Painter, M., Dewitte, P., Blinn, T., and Perakath, B. (1995). Information integration for concurrent engineering (IICE) IDEF3 process description capture method report. Technical report, DTIC Document.

Melnyk, S. A., Lummus, R. R., Vokurka, R. J., Burns, L. J., and Sandor, J. (2009). Mapping the future of supply chain management: A Delphi study. *International Journal of Production Research*, 47(16):4629–4653.

Mentzas, G., Halaris, C., and Kavadias, S. (2001). Modelling business processes with workflow systems: an evaluation of alternative approaches. *International Journal of Information Management*, 21(2):123–135.

Min, H. and Zhou, G. (2002). Supply chain modeling: Past, present and future. *Computers & Industrial Engineering*, 43(1-2):231–249.

Moyaux, T., Chaib-draa, B., and D'Amours, S. (2006). Supply chain management and multiagent systems: An overview. In Chaib-draa, B. and Mller, J., editors, *Multiagent based Supply Chain Management*, pages 1–27. Springer Berlin Heidelberg.

Naim, M., Childerhouse, P., Disney, S., and Towill, D. (2002). A supply chain diagnostic methodology: Determining the vector of change. *Computers & Industrial Engineering*, 43(12):135–157.

OASIS (2007). Web Services Business Process Execution Language v2.0.

OMG (2004). UML 2.0 Superstructure Specification.

OMG (2006). Object Constraint Language v2.0.

OMG (2008). Semantics of Business Vocabulary and Business Rules Specification.

OMG (2011). Business Process Model and Notation (BPMN) v2.0.

Persson, F. (2011). SCOR template – a simulation based dynamic supply chain analysis tool. *International Journal of Production Economics*, 131(1):288–294.

Persson, F. and Araldi, M. (2009). The development of a dynamic supply chain analysis tool – integration of SCOR and discrete event simulation. *International Journal of Production Economics*, 121(2):574–583.

Phelps, R., Parsons, D., and Siprelle, A. (2001). SDI supply chain builder: simulation from atoms to the enterprise. In Peters, B., Smith, J., Medeiros, D., and Rohrer, M., editors, *Proceedings of the 2001 Winter Simulation Conference*, pages 246–249.

Phelps, R. A., Parsons, D. J., and Siprelle, A. J. (2000). SDI industry product suite: simulation from the production line to the supply chain. In Joines, J., Barton, R., Kang, K., and Fishwick, P., editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 208–214.

Piramuthu, S. (2005). Knowledge-based framework for automated dynamic supply chain configuration. *European Journal of Operational Research*, 165(1):219–230.

Prieto-Díaz, R. (1993). Status report: Software reusability. IEEE Software, 10(3):61–66.

Reid, R. D. and Sanders, N. R. (2002). *Operations Management*. John Wiley & Sons, New York, USA.

Ren, C., He, M., Wang, Q., Shao, B., and Dong, J. (2010). Driving supply chain transformation through a business process oriented approach. *Service Science*, 2(4):298–314.

Riddalls, C. E., Bennett, S., and Tipi, N. S. (2000). Modelling the dynamics of supply chains. *International Journal of Systems Science*, 31(8):969–976.

Robertson, D., Bundy, A., Muetzelfeldt, R., Haggith, M., and Uschold, M. (1991). *Eco-Logic: Logic-Based Approaches to Ecological Modelling*. MIT Press, Cambridge, MA, USA.

Ross, R. G. (2003). *Principles of the Business Rule Approach*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ, 2nd edition.

Sadeh, N. M., Hildum, D. W., Kjenstad, D., and Tseng, A. (2001). MASCOT: An agent-based architecture for dynamic supply chain creation and coordination in the internet economy. *Production Planning & Control*, 12(3):212–223.

Sengupta, K., Heiser, D. R., and Cook, L. S. (2006). Manufacturing and service supply chain performance: A comparative analysis. *Journal of Supply Chain Management*, 42(4):4–15.

Shortliffe, E. (1976). *Computer-based medical consultations: MYCIN*. Elsevier, New York.

Simatupang, T., Wright, A., and Sridharan, R. (2002). The knowledge of coordination for supply chain integration. *Business Process Management Journal*, 8(3):289–308.

Siprelle, A., Parsons, D., and Clark, R. (2003). Benefits of using a supply chain simulation tool to study inventory allocation. In Chick, S., Sánchez, P., Ferrin,

D., and Morrice, D., editors, *Proceedings of the 2003 Winter Simulation Conference*, pages 238–245.

Srivastava, S. K. (2007). Green supply-chain management: A state-of-the-art literature review. *International Journal of Management Reviews*, 9(1):53–80.

Stefanovic, D., Stefanovic, N., and Radenkovic, B. (2009). Supply network modelling and simulation methodology. *Simulation Modelling Practice and Theory*, 17(4):743–766.

Stefik, M. (1995). *Introduction to Knowledge Systems*. Morgan Kaufmann, San Francisco, CA, USA.

Sterman, J. D. (1989). Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment. *Management Science*, 35(3):321–339.

Stock, J., Boyer, S., and Harmon, T. (2010). Research opportunities in supply chain management. *Journal of the Academy of Marketing Science*, 38(1):32–41.

Storey, J., Emberson, C., Godsell, J., and Harrison, A. (2006). Supply chain management: theory, practice and future challenges. *International Journal of Operations & Production Management*, 26(7):754–774.

Supply Chain Council (2008). Supply Chain Operations Reference Model, Version 9.0.

Swaminathan, J. M., Smith, S. F., and Sadeh, N. M. (1998). Modeling supply chain dynamics: A multiagent approach. *Decision Sciences*, 29(3):607–632.

Tan, K. C. (2001). A framework of supply chain management literature. *European Journal of Purchasing & Supply Management*, 7(1):39–48.

Tang, C. S. (2006). Perspectives in supply chain risk management. *International Journal of Production Economics*, 103(2):451–488.

Tarantilis, C., Kiranoudis, C., and Theodorakopoulos, N. (2008). A web-based ERP system for business services and supply chain management: Application to real-world process scheduling. *European Journal of Operational Research*, 187(3):1310–1326.

Terzi, S. and Cavalieri, S. (2004). Simulation in the supply chain context: A survey. *Computers in Industry*, 53(1):3–16.

The Business Rules Group (2000). Defining business rules – what are they really?

Trkman, P., Stemberger, M., Jaklic, J., and Groznik, A. (2007). Process approach to supply chain integration. *Supply Chain Management: An International Journal*, 12(2):116–128.

Umeda, S. and Zhang, F. (2006). Supply chain simulation: Generic models and application examples. *Production Planning & Control*, 17(2):155–166.

van der Aalst, W. (1998). The application of Petri Nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66.

van der Aalst, W. and ter Hofstede, A. (2005). YAWL: yet another workflow language. *Information Systems*, 30(4):245–275.

van der Aalst, W. and van Hee, K. (2004). *Workflow Management: Models, Methods, and Systems*. MIT Press, Cambridge, MA, USA.

van der Vorst, J. G., Tromp, S.-O., and van der Zee, D.-J. (2009). Simulation modelling for food supply chain redesign; integrated decision making on product quality, sustainability and logistics. *International Journal of Production Research*, 47(23):6611–6631.

van der Zee, D. J. and van der Vorst, J. G. A. J. (2005). A modeling framework for supply chain simulation: Opportunities for improved decision making. *Decision Sciences*, 36(1):65–95.

Venkatasubramanian, V., Rengaswamy, R., and Kavuri, S. N. (2003). A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies. *Computers & Chemical Engineering*, 27(3):313–326.

Vokurka, R., Choobineh, J., and Vadi, L. (1996). A prototype expert system for the evaluation and selection of potential suppliers. *International Journal of Operations & Production Management*, 16(12):106–127.

Wang, W. Y., Chan, H., and Pauleen, D. J. (2010). Aligning business process reengineering in implementing global supply chain systems by the SCOR model. *International Journal of Production Research*, 48(19):5647–5669.

Wooldridge, M. J. (2001). *Introduction to MultiAgent Systems*. John Wiley & Sons, Inc., New York, NY, USA.

Wu, T., Blackhurst, J., and O'Grady, P. (2007). Methodology for supply chain disruption analysis. *International Journal of Production Research*, 45(7):1665–1682.

Yourdon, E. and Constantine, L. L. (1979). *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Prentice Hall, Upper Saddle River, NJ, USA, 1st edition.

Zarandi, M. F., Pourakbar, M., and Turksen, I. (2008). A fuzzy agent-based model for reduction of bullwhip effect in supply chain systems. *Expert Systems with Applications*, 34(3):1680–1691.

Zegordi, S. H. and Davarzani, H. (2012). Developing a supply chain disruption analysis model: Application of colored Petri-nets. *Expert Systems with Applications*, 39(2):2102–2111.

Appendix A

Experiment Questionnaire

Question 1

Supplier1's process instance bpm-362/sup1_d13 of type D1.3 ("Reserve Inventory") has a longer duration than normally (i.e. 3 instead of 1).

What are the direct and indirect effects of this situation on any SC member and the SC as a whole?

How many chips would you like to bet?

0 …… **1** …… **2** …… **3** …… **4** ……

Question 2

Supplier1's process instance bpm-35/sup1_m16 of type M1.6 ("Release Product") has a longer duration than normally (i.e. 2 instead of 1).

What are the direct and indirect effects of this situation on any SC member and the SC as a whole?

How many chips would you like to bet?

0 …… **1** …… **2** …… **3** …… **4** ……

Question 3

The on-time rate of Manufacturer is lower than expected.

Identify all root causes of this situation.

How many chips would you like to bet?

0 …… **1** …… **2** …… **3** …… **4** ……

Question 4

Supplier2 tracks the received order 274 (for 24 product2 placed by Supplier4) as unusually big through its process instance bpm-275/sup2_d12. We also have that Supplier4 has a low on-time rate.

Does the first situation (at Supplier2) cause the second one (at Supplier4)?

How many chips would you like to bet?

0 …… **1** …… **2** …… **3** …… **4** ……

Appendix B

Questions for User-based Evaluation

Part A

In the scenarios that were used for this experiment there were three main types of problems: delays, demand discrepancies and errors with products.

How common are delays in a Supply Chain, in your opinion?

1 …… **2** …… **3** …… **4** …… **5** ……

How important is the effect of delays on Supply Chain Performance, in your opinion?

1 …… **2** …… **3** …… **4** …… **5** ……

How common are demand discrepancies in a Supply Chain, in your opinion?

1 …… **2** …… **3** …… **4** …… **5** ……

How important is the effect of demand discrepancies on Supply Chain Performance, in your opinion?

1 …… **2** …… **3** …… **4** …… **5** ……

How common are errors with products in a Supply Chain, in your opinion?

1 …… **2** …… **3** …… **4** …… **5** ……

How important is the effect of errors with products on Supply Chain Performance, in your opinion?

1 …… **2** …… **3** …… **4** …… **5** ……

Part B

In one of the scenarios you were asked to answer questions by utilising the explanation provided by the tool.

How useful did you find the explanation provided by the tool, as opposed to answering questions without such support?

1 …… **2** …… **3** …… **4** …… **5** ……